Bevezetés a MATLAB® használatába
Híradástechnika és Jelfeldolgozás

Wührl Tibor

Tartalom

Bevezetés

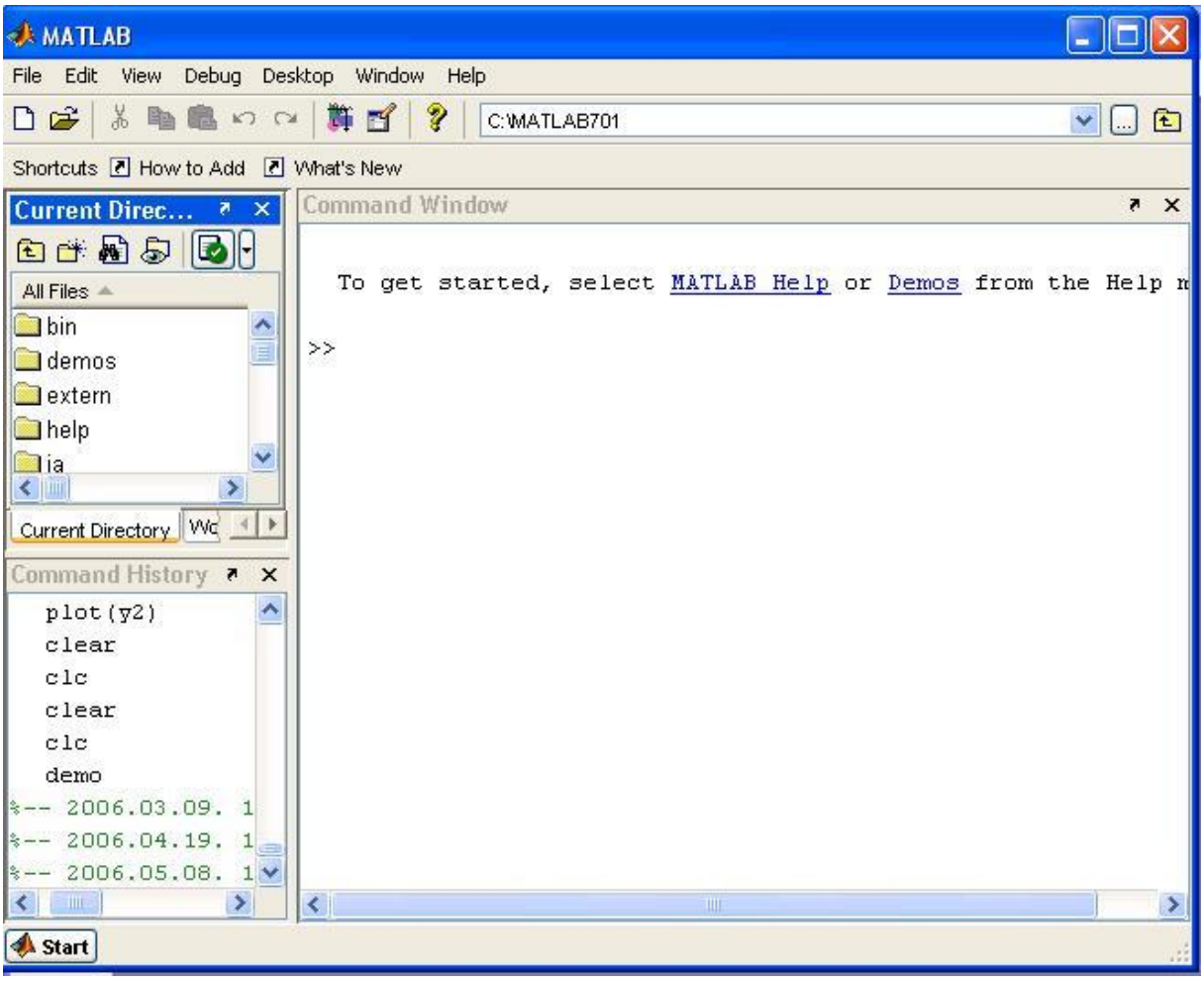
Kedves Olvasó!
Még mielőtt nekilátnánk a munkának, beszéljük meg pontosan, hogy mi is a MATLAB®!
Természetesen egy programcsomagról van szó, melyet a MathWorks, In.. készített, és folyamatosan fejleszt. A MATLAB® név a MATrix LABoratory-ból ered.
Mire is jó ez?
A MATLAB® egy olyan matematikai programcsomag, melyet elsősorban mérnököknek fejlesztettek, de mára már a biológiában, gazdasági számításokban is nagy sikerrel alkalmazzák. Jelen jegyzet elsősorban villamosmérnök szemléletmódban kívánja tárgyalni a MATLAB®-ot. A felhasználási példák és gyakorlati feladatok bemutatása elsősorban híradástechnikai- és jelfeldolgozási problémák megoldására összpontosít. A MATLAB® UNIX® és MS-WINDOWS® platformon futtatható. A munkát minden esetben jól konstruált „HELP" segíti. A „HELP"-ben és a „DEMO"-ban bemutatott mintapéldák újabb alkalmazások és függvények megismerését segítik elő. Hasznos és naprakész tartalomhoz juthatunk a www.mathworks.com WEB oldalon. Regisztrált felhasználók WEB-es szemináriumokon, úgynevezett „WEBINAR"-on vehetnek részt, melyek segítségével a tudás egy adott témakörben mindig naprakész lehet.
Jelen jegyzetben a célom az, hogy egyszerűen és érthetően bemutassam a MATLAB® használatát, ezért az egyes fejezetekben egyszerű példákkal találkozhat a Kedves Olvasó.

Eredményes, kitartó munkát kívánok!

A szerző

Kezdjünk neki egyszerűen!

Indítsuk el a MATLAB®-ot a matlab.exe futtatásával, vagy a MATLAB® ikonra kattintással. Ezután nyugodtan „eldobhatjuk” a papírt, ceruzát és a zsebszámológépet, mert számítási feladatainkat a továbbiakban egyszerűbben tudjuk elvégezni ebben a munkakörnyezetben. A program indítása után a képernyőn következőt (vagy valami hasonlót) látjuk:



1.1. ábra MATLAB® kezdő képernyő

A felső sorokban a már jól megszokott menüpontokat és az egyes funkciók gyors elérését lehetővé tevő ikonok sorát láthatjuk. Az ikon sor alatti képernyő terület alapbeállításban három részre osztott „Current Directory”, „Command Window” és „Command History”. A Command Window, - amit a továbbiakban parancs ablaknak nevezünk – lehetőséget ad a MATLAB®parancsok közvetlen bevitelére. A parancsokat a >> prompt jel után kell írnunk, minden parancs sort ENTER billentyűleütéssel zárunk. A korábban a parancs ablakba beírtak a Command History ablakból visszakereshetők, újrafuttathatók. Abban az esetben, ha a Kedves Olvasó gyönyörködni kíván a MATLAB® képességeiben, kattintson a parancs ablak „demos” pontjára, de azt javaslom, hogy ezt majd később tegyük, akkor, ha már kellemesen elfáradtunk és lazítani kívánunk.

Most nézzük a következő egyszerű feladatot és annak megoldását!

$$1 + 2 = 3 \qquad (1.1.)$$

Hogy lehet mindezt MATLAB®-bal megoldani? Egyszerűen! Írjuk a parancs ablakba a következőt, majd üssünk ENTER-t!

```
>> 1+2

ans =

     3
```
(1.2.)

Válaszként az „ans = 3” –ot kaptuk, ami a várt eredménnyel egyező.

A következő példánál is maradjunk az előző (1.1.) egyenlőségnél, de most már bonyolítsunk a helyzeten, és a feladatot fogalmazzuk meg általánosabban, például használjuk a:

$$c = a + b \qquad (1.3.)$$

kifejezést!
Ekkor „a” és „b” konstanst (persze ami a továbbiakban változó is lehet) kell deklarálnunk. Ez most így bonyolultnak tűnik, de nézzük milyen egyszerű!

```
>> a=1;
>> b=2;
```
(1.4.)

A sort most pontosvessző karakterrel zártuk (majd utána természetesen ENTER – de ez ugye nem látszik), ebben az esetben az aktuális értéket eltárolja a MATLAB, de azt a képernyőn nem jeleníti meg. Pontosvessző karakter nélkül a következőt látnánk:

```
>> a=1

a =

     1
```
(1.5.)

A (4.) kifejezéssel deklaráltuk „a” és „b” értékét, melyekkel most már szabadon végezhetünk matematikai műveleteket. A >> prompt jel után írhatjuk:

```
>> c=a+b;
```
(1.6.)

Rendben! De most hol az eredmény? A pontosvessző karakter miatt nem jelent meg. Sok esetben hasznos, és az áttekinthetőséget könnyíti, ha az egyes részeredmények nem jelennek meg. A „c” változónk persze megkapta a helyes értékét, melyet a (1.6.) összefüggéssel kalkuláltunk, de azt most csak a számítógépünk memóriájában tároltuk, a képernyőn nem jelenítettük meg. Az eltárolt eredmények, részeredmények természetesen bármikor lekérdezhetőek, megjeleníthetőek. Ekkor egyszerűen a >> promt után a változó nevét kell leírnunk, majd ENTER-t ütünk, például:

```
>> c

c =
```

3 (1.7.)

A számítógép memóriájában tárolt konstansok és változók, valamint részeredmények nevére nem mindig emlékszünk. A MATLAB® „who” parancs lehetőséget ad a deklarált változók megjelenítésére:

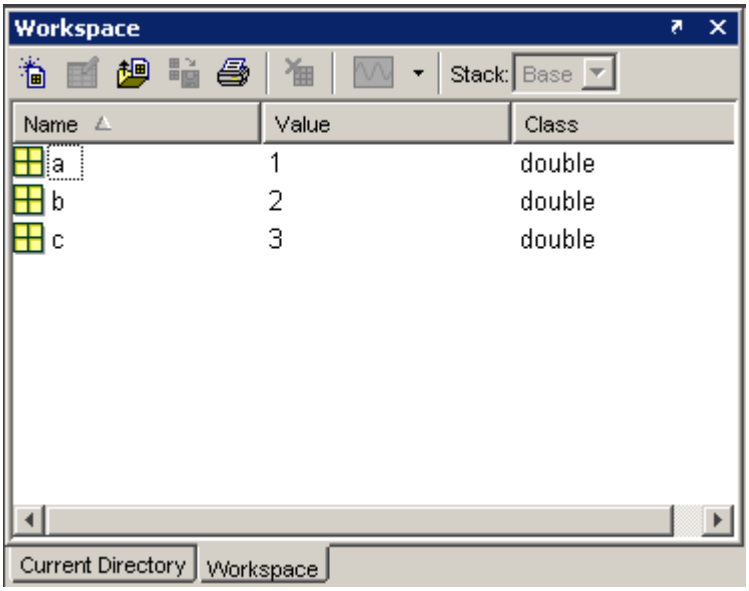
```
>> who

Your variables are:

a   ans  b   c
```
(1.8.)

A lekért változók neveit ABC sorrendben láthatjuk, melyek: „a”, „ans”, „b” és „c”.
Az egyes változók aktuális értéke a (1.7.) szerint jeleníthető meg. Az „ans” változót akkor hoztuk létre, amikor olyan eredményt (részeredményt) kalkuláltunk, melynek korábban nem deklaráltunk változót (1.2.).

A „whos” parancs kiadásával a változók tulajdonságait is megjeleníthetjük:

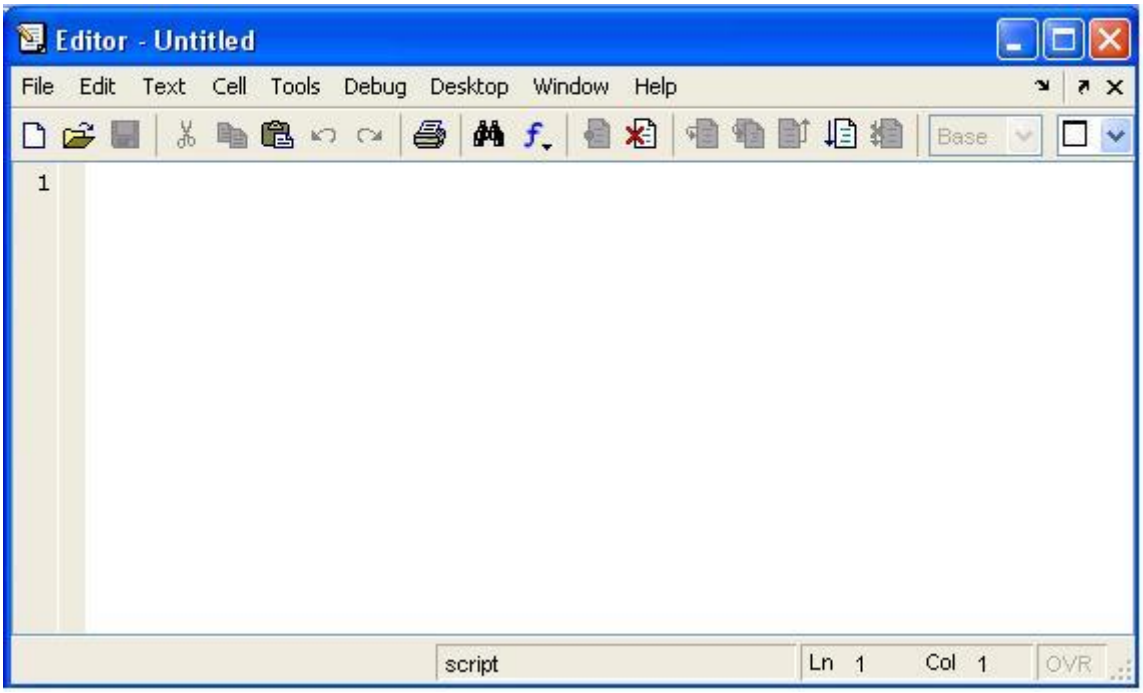
```
>> whos
  Name  Size            Bytes  Class

  a     1x1                 8  double array
  b     1x1                 8  double array
  c     1x1                 8  double array

Grand total is 3 elements using 24 bytes
```

Az itt kapott információk alapján látható, hogy a változóink egy 1x1-es mátrixba tároltak, mivel „sima” számokról van szó, a változó típusa double, ami a memóriában így 8 byte helyet foglal.
A változóinkról információkat kaphatunk még a Workspace ablakban a Workspace fülre kattintva is:



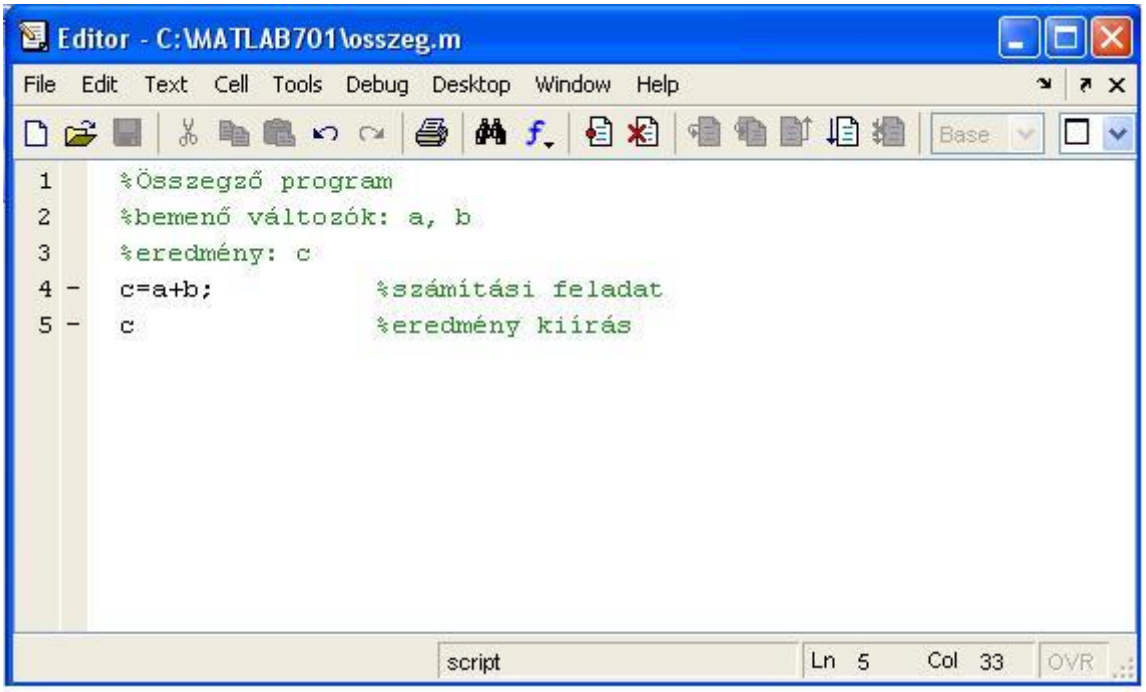
Itt ha duplán kattintunk egy adott változó nevén, a parancsablak helyén megnyílik az Array Editor ablak, melyben szerkeszthetjük, illetve az Excel megjelenéséhez hasonló környezetben láthatjuk illetve módosíthatjuk változóinkat. Ennek használata természetesen majd a mátrixok használata esetén fordul elő gyakrabban.

Összetettebb számítási feladatok elvégzése esetén célszerű a számítási algoritmust fájlként tárolni. A számítási feladatokat tároló fájlok általában az „m" kiterjesztést kapják. Ezek a fájlok közvetlenül a parancs ablakból futtathatók. Maradjunk ez előzőekben tárgyalt egyszerű példánknál, de még mielőtt nekilátnánk, töröljük a memóriából az eddig deklarált változóinkat! Ezt a „clear" parancs beírásával tehetjük meg, majd ENTER-t kell ütnünk. A már előzőekben megismert „who" parancsra (1.8.) most nem kapunk választ, vagyis nincs érvényesen deklarált változónk.
Most, hogy a MATLAB®-unk tiszta lappal indul válasszuk a File menüből a New pontot és azon belül az M-File-t. Ekkor egy szövegszerkesztőt hív be a MATLAB®. A megnyitott dokumentum pedig „Untitled".



1.2. ábra MATLAB® Editor

A szerkesztő program segítségével készítsük el az egyszerű „m" fájlt, melynek az osszeg.m nevet adjuk. A fájl elmentése (névadással) a File menüpont Save as pont választásával történhet. A megírt program a következőképpen néz ki:



1.3. ábra Futtatható „m” fájl

Az Editor program a sorokat automatikusan sorszámmal látja el. A futtatható sorokhoz a sorszám után „-” jelet ír. A megszerkesztett programba megjegyzéseket (komment) is tehetünk. A százalék (%) jel után írt karakterek megjegyzésként tároltak.

Az „m” fájl futtatására több lehetőségünk van, melyek közül a legegyszerűbb a fájl nevének beírása (kiterjesztés nélkül) a parancssorba, majd ENTER. Ekkor csak arra kell ügyelnünk, hogy a MATLAB® elérési útban szereplő könyvtárban, vagy mappában legyen a szerkesztett „m” fájl. (Azaz a Current Directory ablakban alapértelmezettként megjelenített mappába történjen a mentés) A futtatás megoldható közvetlenül az Editor programból is a Debug menü Run pont választással (F5 – funkció billentyű, vagy Run ikonra kattintás).
Természetesen futtatás előtt a bemenő változókat deklarálni kell, különben a következő hibaüzenetet kapjuk:

??? Undefined function or variable "a". (1.9.)

A változók helyes deklarációja után (például (1.4.)), az „m” fájl sikeresen futtatható.

Az „m” fájl készítésére adódik még egy másik praktikus megoldás is: a parancsablakba kiadott parancsaink sorozatát – válogatva is – elmenthetjük „m” fájlba. Az előző példánál maradva, ha a parancsablakban kiadjuk a „c=a+b” majd a „c” parancsot (persze a változókat érdemes előtte megadni, de az sem baj, ha nem) a parancsok bekerülnek a History ablak listájába. Kattintsunk itt a c-re, majd a shift gomb nyomva tartása mellett a „c=a+b” parancsra, hogy mind a kettőt kijelöljük. (CTRL gomb használatával nem egymás utáni parancsokat is kijelölhetünk). A kijelölésen jobb gombot nyomva a megjelenő menüben válasszuk a Create M-file pontot, melynek hatására a fent bemutatott ablak nyílik meg, ezután már csak a mentés (és kommentezés) van hátra.

Most már nyugodtan hátradőlhetünk a fotelban, hiszen megismertük a MATLAB®-ot, és tudjuk is azt használni! Igaz, még egy picit gyakorolnunk kell a használatát, hogy az készség

szintre fejlődjön, valamint célszerű megismerni néhány függvényt és az egyes téma- és tudományterületekhez rendelkezésre álló „szerszámkészletet”, úgynevezett „tool-box”-ot.
A további fejezetekben tehát elsősorban olyan számtanpéldákat oldunk meg, melyek a fenti cél elérését szolgálják.
Kedves Olvasóm! Ha kellemesen elfáradt, most javaslom pihenés és lazításképpen hívja be a MATLAB® demót (parancsablak „demos” pontra kattintással, mely a lap tetején található, vagy a >> prompt után gépeljük be a „demos” szót, majd üssünk ENTER-t.).

Számábrázolás

Az előző fejezetben a cél az volt, hogy megismerjük a MATLAB®-ot, ezért valós egész számokkal dolgoztunk. Természetesen a MATLAB® ennél sokkal többre képes! Az egész számok megjelenítését és deklarációját már megismerhettük. A tört számokat megadhatjuk tizedes tört alakban. Az egész részt és a tört részt tizedes pont választja el, ami a következőképpen néz ki:

```
>> a=3.456

a =

   3.4560                                   (2.1.)
```

Megadhatjuk a törtszámot emeletes tört alakban is a számláló/nevező deklarálásával, például:

```
>> a=3456/1000

a =

   3.4560                                   (2.2.)
```

Az adattárolás ekkor is az előzővel megegyező tizedes tört alakban történik.
Az ábrázolás formátum a „format + paraméter” paranccsal adható meg.
A fenti példákban (2.1.) (2.2.) a „format short” beállítás volt érvényes. Pontosabb számábrázolást és megjelenítést tesz lehetővé, ha átváltunk „format long”-ra.
A „format long”-ra váltást követően kérdezzük le a π értékét. {A π előre definiált konstans, melyre a „pi”-ként hivatkozhatunk.}

```
>> format long
>> pi

ans =

   3.14159265358979                         (2.3.)
```

Az ábrázolás most 14 tizedesig történt.
A formátumok csak felsorolás szinten a következők:

Parancs:	Jelentése:
format short	Megjelenítés 4 tizedesig

format long	Megjelenítés 14 tizedesig
format shotr e	Megjelenítés lebegő pontos alakban 4 tizedesig, Például: 1.2345e+002 Ha a hatványkitevő 0, akkor 000-t jelenít meg.
format short g	Megjelenítés lebegő pontos alakban 4 tizedesig, Például: 1.2345e+002 Ha a hatványkitevő 0, akkor az exponenciális tagot nem jeleníti meg.
format long e	Megjelenítés lebegő pontos alakban 14 tizedesig, Például: 1.23454455667788e+002 Ha a hatványkitevő 0, akkor 000-t jelenít meg.
format long g	Megjelenítés lebegő pontos alakban 14 tizedesig, Például: 1.23454455667788e+002 Ha a hatványkitevő 0, akkor az exponenciális tagot nem jeleníti meg.
format bank	Megjelenítés 2 tizedesig
format rat	Megjelenítés közönséges tört alakban. A számláló és a nevező a / jellel elválasztott.
format hex	Megjelenítés hexadecimális alakban.
format compact	Megjelenítéskor elhagy minden üres sort.

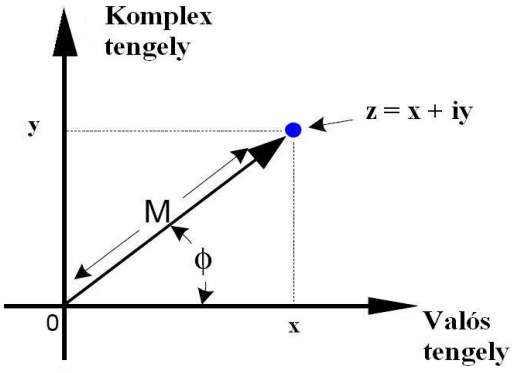
A felsorolt és alkalmazható formátumok pontos tulajdonságait a „help format” paranccsal kérhetjük le.

Műszaki feladatok megoldása esetén gyakran szükséges a komplex szám ábrázolása, valamint műveletvégzés a komplex számokkal. Komplex számokat is egyszerűen deklarálhatunk a MATLAB®-ban. Mielőtt megnéznénk a komplex számok deklarációját és a velük végezhető műveleteket, tekintsük át ismétlésként a komplex számokat és azok jelentését.

A komplex számok valós és képzetes részből állnak. A képzetes részt „j” vagy az „i” operátorral képezzük (MATLAB® mindkét operátor jelölést elfogadja, de a „j”-t „i”-nek konvertálja). A továbbiakban a komplex operátort „i”-vel jelölöm, ami nem más mint a -1 szám gyöke, vagyis $i^2 = -1$. {Karl Gauss a „j”-t az árnyék árnyékának nevezte – „shadow of shadows”}
A komplex szám tehát két tagból, valós és képzetes részből áll:

$$z = x + yi \qquad (2.4.)$$

A „z” komplex számot ábrázolhatjuk derékszögű koordináta rendszerben. Ekkor választhatjuk például a vízszintes tengelyt valós tengelynek, a függőlegeset pedig képzetesnek:



2.1. ábra z = x + iy komplex szám

A 2.1 ábrán látható komplex szám jellemezhető továbbá azzal a vektorral is, mely az origóból a komplex számra mutat. A vektor megadható a valós tengely és a vektor által bezárt szöggel {Φ}, valamint a vektor hosszával {M}.
Ezen vektor paraméterek ismeretében a komplex szám megadható a következő alakban, úgynevezett polár formulával:

$$z = e^{j\Phi} \qquad (2.5.)$$

A vektor hossza:

$$M^2 = x^2 + y^2 \qquad (2.6.)$$

összefüggésből számolható, míg a vektor állása:

$$\Phi = \text{atan}\,(y / x) \qquad (2.7.)$$

összefüggéssel adható meg. Az összefüggések egyszerűen beláthatók a 2.1 ábra segítségével.
A komplex számot gyakran a trigonometrikus alakban adjuk meg:

$$z = M(\cos\Phi + i \sin\Phi) \qquad (2.8.)$$

Most, hogy átismételtük a komplex számok ábrázolás módját, definiáljunk két komplex számot az alábbiakban:

```
>> a=1+3i;
>> b=2-0.5i;                    (2.9.)
```

Kérdezzünk rá a „b” számra a szokásos módon! Ekkor a következő eredményt kapjuk:

```
>> b

b =

   2.0000 - 0.5000i             (2.10.)
```

Most adjuk össze a két komplex számot. Az összegzéshez használhatjuk az előző fejezetben megírt „m” fájlt (1.3 ábra). Az eredmény ekkor a „c” változóban tárolt:

```
c =

   3.0000 + 2.5000i                    (2.11.)
```

A fentiekből látható, hogy a komplex számokkal a műveleteket ugyan úgy kell végeznünk (MATLAB-ban kijelölnünk), mint azt a valós számok esetén tettük. A komplex szám algebrai alakjának ismeretében a polár vagy a trigonometrikus alak az ismert összefüggésekkel (2.6.) (2.7.) meghatározható, a MATLAB® segédfüggvényekkel könnyíti meg dolgunkat. A komplex szám abszolút értéke az „abs", szöge pedig az „angle" függvénnyel számolható. Az „angle" függvény az eredményt radiánban adja $\pm\pi$ tartományban. Példaként számoljuk ki az előző komplex eredmény (2.11) abszolút értékét és szögét:

```
>> abs(c)

ans =

   3.9051

>> angle(c)

ans =

   0.6947                              (2.12.)
```

Természetesen a fenti számítások úgy is elvégezhetők, hogy a kapott eredményeket deklarált változókban tároljuk, ugyan úgy, mint azt az (1.6) példában is tettük.

Egy változó deklarálásánál a következőket kell összefoglalnunk:

- A változó típusát előre nem kell ismernünk, valamint nem kell megadnunk.
- A változó deklarálásánál célszerű kerülnünk a MATLAB® belső deklarációra felhasznált karaktereket, például: i, j, pi stb, valamint a speciális karaktereket.
- Műveleti jelek és ékezetes betűk nem szerepelhetnek a változó nevében.
- Változók deklarálásához több karakterből álló sztringet is használhatunk (maximum 31 karakterből állhat egy deklarált változó), azok akár szavak is lehetnek.
- A MATLAB® a kis és nagybetű között különbséget tesz, tehát egy deklarált „A" változó nem megegyező az „a" változóval.

Vektorok és mátrixok

Az előző fejezetben áttekintettük a valós és a képzetes számok ábrázolását. A skalár számok vektorokba, mátrixokba rendezhetők, sőt a skalár számot felfoghatjuk úgy is, mint egy **1**x**1**-es mátrixot, valamint egy **n** elemből álló vektort is, mint egy **n**x**1** vagy **1**x**n** méretű mátrixot.
A mátrix és vektor definiáláskor a mátrix elemeket, egymástól vesszővel kell elválasztanunk, az egyes sorhatárok esetén pedig elválasztó jelnek a pontosvesszőt kell alkalmaznunk. Az elemek közé célszerű egy-egy szóköz (space) karaktert is beszúrni. Az elemeket szögletes zárójelek [] közé kell írnunk.
A mátrix definiálás példában, az úgynevezett Dürer mágikus mátrixot fogjuk alkalmazni. A mátrix egy reneszánsz rézkarcon (Melencolia I) található (Albrecht Dürer).
Definiáljuk most ezt a mátrixot a MATLAB®-ban, a mátrixot jelöljük A-val!

```
>> A=[16, 3, 2, 13; 5, 10, 11, 8; 9, 6, 7, 12; 4, 15, 14, 1];
```
(3.1.)

Most kérdezzük le az „A” változónk értékét a szokásos módon!

```
>> A
A =
   16    3    2   13
    5   10   11    8
    9    6    7   12
    4   15   14    1
```
(3.2.)

A továbbiakban ezen a mátrixon mutatunk be néhány egyszerű mátrix műveletet. A műveletvégzés kapcsán fogjuk megtudni, hogy miért is nevezzük a fenti mátrixot mágikusnak.

Képezzük a mátrix oszlop elemeinek összegét. Jelen méretű mátrixnál persze ez ránézésre megadható, nem sokkal bonyolultabb feladat, mint az (1.1.) volt.

```
>> sum(A)

ans =

   34   34   34   34
```
(3.3.)

Látható, hogy minden oszlop elemeinek összege 34-nek adódik.
Most végezzük el az összegzést soronként is!

```
>> sum(A')'

ans =

   34
   34
   34
   34
```
(3.4.)

Az összeg képzésnél egy kis trükköt alkalmaztunk, ugyanis képeztük a Dürer mátrix transzponáltját (vagyis a mátrix oszlopait és sorait felcseréltük, a mátrix soraiból oszlopokat, az oszlopokból pedig sorokat képzünk). A transzponálást a ’ operátorral végeztük. Az így kapott transzponált mátrix oszlopainak elemét összegeztük és a kapott sorvektort is transzponáltuk, így egy négy elemű oszlopvektort kaptunk. Az A mátrix transzponáltja a következőképpen néz ki:

```
>> A'

ans =

   16    5    9    4
    3   10    6   15
    2   11    7   14
```

13 8 12 1 (3.5.)

Most nézzünk egy további műveletet, emeljük ki egy vektorba a fő átló elemeit! Ezt a „diag” függvénnyel tehetjük meg:

```
>> diag(A)

ans =

   16
   10
    7
    1
```
(3.6.)

A főátló kiemelés eredménye egy oszlopvektor. Ha ránézünk az oszlopvektorra, akkor rögtön észrevehető, hogy az oszlopvektor elemeinek összege ismét 34 lesz. Most ellenőrizzük a MATLAB-bal, hogy a meglátásunk igaz-e:

```
>> sum(diag(A))

ans =

   34
```
(3.7.)

Valóban! Az eredmény megegyezik azzal amit vártunk, vagyis a mágikus Dürer mátrix főátlóban szereplő tagokat összegezzük, akkor megegyező számot kapunk a mátrix soraiból képzett vektor elemek összegével, valamint az oszlop elemek összegével is. A (3.7.) műveletben természetesen nem az eredmény (34) okozza a legnagyobb örömet, hanem az, hogy az egyes függvények egymásba ágyazhatók! A MATLAB® tehát lehetőséget ad az egyes függvények egymásba ágyazására, így bonyolult matematikai kifejezéseket hozhatunk létre anélkül, hogy közbenső változóban kellene az egyes részeredményeket tárolni. Természetesen előfordulhat olyan eset, hogy a részeredmény is hordoz hasznos információkat (például hibakeresésnél), ekkor persze ezen részeredményekhez is kell (célszerű) változót deklarálni.

A fenti példákban megismertük, hogy miként lehet mátrixot deklarálni, elemeit megadni és ki is próbáltunk néhány mátrix manipulatív műveletet. Igazi műszaki jelentése akkor van a mátrixoknak, ha azok segítségével például egyenletrendszert tudunk megoldani. Ehhez viszont a fentiekben kipróbált műveletek nem elégségesek.
A továbbiakban megvizsgálunk néhány további hasznos műveletet.

A számítások során gyakran szükséges egy mátrix, vagy vektor elem kiemelése. Például egy adott elemel, mint skalár mennyiséggel kívánjuk folytatni a számításainkat.
Példánkban ez az elem kiemelés legyen a Dürer mátrix második sor harmadik oszlop eleme. Ekkor a következőt kell tennünk:

```
>> elem23=A(2,3)

elem23 =
```

Az A-val jelölt mátrix második sorának harmadik elemét az „elem23” változóba mentettük.

Részmátrixok kiemelése, a „:” paraméter.
Az elemre hivatkozás parancsát kicsi kiegészítéssel használhatjuk részmátrixok kiemelésére is. Az A mátrix bal felső sarkában elhelyezkedő 4 db számot kiemelhetjük egy új változóba a Aresz=A(1:2,1:2) paranccsal. így az Aresz tartalma a következő lesz:

Aresz =

```
   16    3
    5   10
```

(A „:” jelentése itt kb. „tól-ig”-nak felel meg). Ha önmagában használjuk akkor az a teljes elérhető sort/oszlopot fogja jelenteni: így a Aelsosor=A(1,:) (esetünkben egyenértékű az A(1,1:4) paranccsal) eredménye a következő:

Mátrixok tükrözése, forgatása:
flipud(A) parancs a mátrixot vízszintes tengely mentén, (fentről-le – up-down) , a fliplr(A) függőleges tengely mentén tükrözi (left-right), míg a rot90(A) parancs 90 fokkal az óramutató járásával ellentétes irányba forgatja a mátrixot.

A lineáris algebrában nagy jelentősége van a mátrix determináns számításnak. Négyszer négyes mátrix esetén még jó esélyünk van a determináns érték meghatározásához, de már az is időrabló feladat, ha a számítást papíron, ceruzával végezzük. A kézi számolás sok hibalehetőséget is rejt magában, nagyobb méretű mátrixok esetén pedig szinte lehetetlen papíron megoldani a feladatot.
A Kedves Olvasót arra bíztattam mindjárt az első fejezet elején, hogy dobjuk el a papírt és a ceruzát! Most is ezt teszem! Nem szabad ugyanakkor megfeledkeznünk arról sem, hogy a számítógép csak a papír és a ceruza funkciót látja el, helyettünk a számítógép nem gondolkozik, ezért nekünk mindig pontosan tudnunk kell, hogy melyek az egyes műveletek elvégzésének szabályai.
Számítsuk ki az A mátrix (3.2.) determinánsát a „det” függvénnyel:

```
>> det(A)

ans =

     0
```

(3.9.)

A Dürer mátrix tehát szinguláris, a determináns értéke nulla.
Matematikai ismereteinket, ha felidézzük, akkor tudjuk, hogy szinguláris mátrixnak nincs inverze. Próbáljuk meg mégis kiszámítani az A mátrix (3.2.) inverzét. Használjuk az „inv” függvényt:

```
>> inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 4.383047e-018.
```

```
ans =

  1.0e+015 *

    0.2796    0.8388   -0.8388   -0.2796
   -0.8388   -2.5164    2.5164    0.8388
    0.8388    2.5164   -2.5164   -0.8388
   -0.2796   -0.8388    0.8388    0.2796
```
(3.10.)

Nem egészen a várt eredményt kaptuk, vagy mégis? Tudnunk kell, hogy a számítógépünk számábrázolási pontossága véges, ezért sok esetben közelítő eredményt kapunk. A kapott mátrix minden tagját szoroznunk kell 10^{15}-el, vagyis irreálisan nagy értékű elemekből áll a számított mátrix. Ez arra vezethető vissza, hogy kerekítési okokból nullával osztás helyett az osztás valamilyen kis számmal (nullához közeli) történt. A figyelmeztető üzenet (Warning) jelzi számunkra azt, hogy a kapott eredmény nincs teljesen rendben. Az eredmény értékelésénél az ilyen üzeneteket minden esetben figyelembe kell vennünk!

A további műveletek megismeréséhez már nem a mágikus Dürer mátrixot fogjuk használni. Nézzünk egy egyszerű, gyakorlati feladatot (ismét nem a bonyolult számtanpélda kiválasztás volt a cél, hanem az, hogy a számítások átláthatók, ellenőrizhetők legyenek):

Oldjuk meg az

$$\begin{array}{llll} 5x & + & 3y & = & 57 \\ 10x & - & 2y & = & 42 \end{array} \qquad (3.11.)$$

egyenletrendszert!
Az egyenletrendszert felírhatjuk mátrix alakban:

$$\begin{bmatrix} 5 & 3 \\ 10 & -2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 57 \\ 42 \end{bmatrix}$$

(3.12.)

A mátrixos alakból látszik, hogy az x és y elemből álló vektor értékéhez úgy juthatunk, ha az 5, 3, 10, -2 elemek alkotta mátrix inverzét szorozzuk az 57 és 42 elemekből álló oszlopvektorral.
MATLAB-ban a feladatot egyszerűen megoldhatjuk:

```
>> A=[5, 3; 10, -2]
A =

     5     3
    10    -2

>> b=[57, 42]'

b =
```

```
    57
    42

>> c=inv(A)*b

c =
     6
     9                                                    (3.13.)
```

Az eredményt a **c** oszlopvektor elemei adják, vagyis x = 6, míg y = 9 eredményhez jutottunk. Visszahelyettesítve a kiinduló egyenletbe (3.11.), láthatjuk, hogy az eredményünk helyes.
A fenti példában tehát egy lineáris egyenletrendszer hatékony megoldását láthattuk. Ezt a módszert eredményesen használhatjuk lineáris áramkörök analíziséhez.

A fenti példában (3.13.) mátrix szorzását végeztük el egy oszlopvektorral. Értelmezhető művelet továbbá a mátrixok összeadása és kivonása is, ekkor a szokásos „+” vagy „-” operátort kell alkalmaznunk. Elfordulhat olyan eset is, hogy a műveleteket elemenként kell elvégeznünk. A mátrix, vagy vektor elemeire hivatkozást az előzőekben már láthattunk (3.8.), de az ott megismertek elsősorban egy elem kiemelését tették lehetővé, tehát az ott megismert eljárással a feladatot igen komplikált lenne megoldani. Elemenkénti művelet végzésre a MATLAB® külön operátort definiált, melyek a következők:

Műveleti jel:	**Jelentése**
□ * (tizedes pont és csillag)	Mátrix vagy vektor elemek elemenkénti szorzása
. / (tizedes pont és per jel)	Mátrix vagy vektor elemek elemenkénti osztása
. ^ (tizedespont és fel-nyíl)	Mátrix vagy vektor elemek elemenkénti hatványozása

Vizsgáljuk meg a szorzás és az elemenkénti szorzás műveletét. A vizsgálatainkhoz definiáljunk egy sor és egy oszlop vektort. Mindkét vektor 5 – 5 elemből álljon. A sorvektort nevezzük „svktor”-nak, az oszlopvektort pedig „ovektor”-nak:

```
>> svektor=[1, 3, -5, 2, 3.65]

svektor =

   1.0000    3.0000   -5.0000    2.0000    3.6500

>> ovektor=[2, 2.51, 4, pi, 2*pi]'

ovektor =

   2.0000
   2.5100
   4.0000
   3.1416
   6.2832                                                (3.14.)
```

Szorozzuk össze a két vektort!

```
>> svektor*ovektor

ans =

  18.7468                                                    (3.15.)
```

A vektoriális szorzat egy skalár számot adott, mely az azonos index számú sor és oszlop elemek szorzatainak összege. A vektor szorzás alapfeltétele az volt, hogy a sorvektor és az oszlopvektor azonos számú elemből álljon.
Elemenkénti szorzás csakis azonos típusú vektorok között történhet, vagyis két azonos elemszámú sorvektorok, vagy azonos elemszámú oszlopvektorok között értelmezhető. Az előzőleg definiált (3.14.) vektorok közül az elemenkénti szorzáshoz az egyik vektorunkat konvertálnunk (vagyis transzponálni) kell:

```
>> svektor.*ovektor'

ans =

   2.0000    7.5300  -20.0000    6.2832   22.9336     (3.16.)
```

A fenti művelet (3.16.) során az oszlopvektort sorvektorrá alakítottuk, így az elemenkénti szorzást két azonos elemszámú sorvektorral végeztük. Az eredmény egy öt elemből álló sorvektor. Természetesen az elemenkénti szorzást elvégezhetjük úgy is, hogy a sorvektort alakítjuk oszlopvektorrá:

```
>> svektor'.*ovektor

ans =

   2.0000
   7.5300
 -20.0000
   6.2832
  22.9336                                                  (3.17.)
```

Az eredmény most egy öt elemű oszlopvektor. Az egyes azonos index számú elemek megegyezők a (3.16.)-ban kapott elemértékekkel.

Az elemenkénti műveletvégzés mátrixok között is értelmezhető, de ne felejtsük el, hogy csakis azonos dimenziójú mátrixok között!

Remélem a Kedves Olvasó most kellemesen elfáradt, de most azt javaslom NE indítsa el a demót, hanem pihenésképpen kezdje el a következő fejezet feldolgozását. Ígérem, a következő fejezet sokkal izgalmasabb lesz, és persze látványos, hiszen „megmozgatjuk” a MATLAB® grafikai lehetőségeit. A száraz számok helyett most szép színes ábrákban gyönyörködhetünk!

Függvények és grafikai lehetőségek

Definiáljunk hamar egy ’fi’ vektort, melyben az elemek 0-tól 2π intervallumban futnak 0.05 lépésközzel! Brrr, az előző fejezet végén nem ebben állapodtunk meg... Türelem, rögtön rajzolunk!
A definiálni kívánt vektor több mint száz elemből áll. Ekkora méretű vektor manuális feltöltése meglehetősen időrabló feladat és sok hibalehetőséget is rejt magában. Vektor definiálásra az eddigiektől eltérő módszert alkalmazunk:

```
>> fi=[0:0.05:2*pi];                    (4.1.)
```

Az így definiált vektor elemeit most nem listázzuk ki, mert a t vektor most több mint 100 elemből áll (pontosan 126). Az elemek értéke:

```
0    0.0500    0.1000  ....    6.2500
```

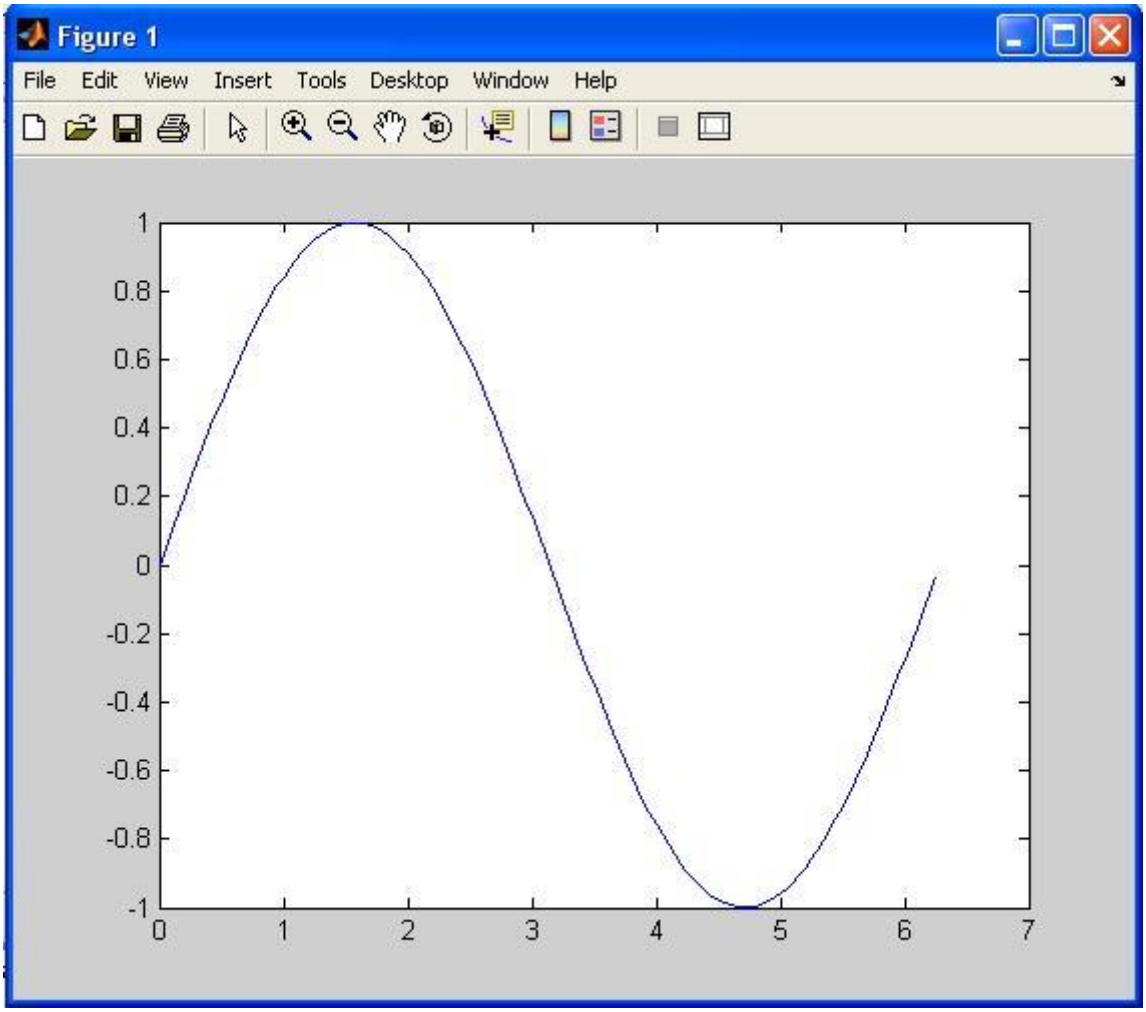
A fi vektor elemeit fogjuk fel úgy, mint ha az, az egységnyi vektor időben változó szöge lenne. Számítsuk ki a vektor vetületét. A számításhoz alkalmazzuk valamely szögfüggvényt, az eredményt szintén tároljuk egy vektorban!

```
a=sin(fi);                           (4.2.)
```

Az a vektor elemeit kilistázhatjuk a szokásos módon, de ezt most ismét nem tesszük (természetesen a Kedves Olvasó megteheti próbaképpen), mert az a vektor a fi vektorral megegyező elemszámú. Természetesen a vektorok szemléletes grafikus megjelenítésére is lehetőségünk van. Adjuk ki a következő parancsot a MATLAB® parancs ablakban:

```
>> plot(fi,a)                           (4.3.)
```

Ekkor a következő ablak jelenik meg:

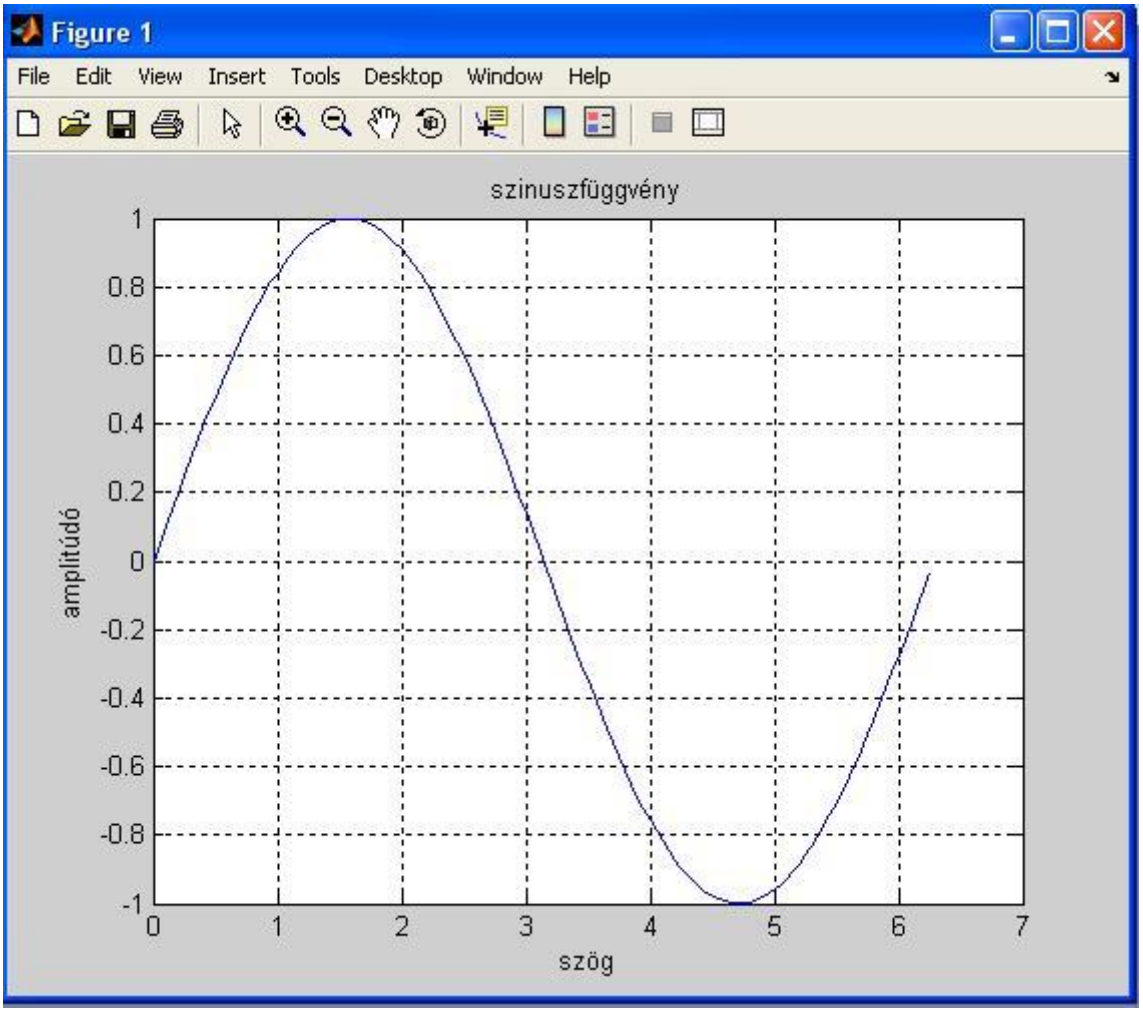


4.1 ábra Szinusz függvény

Az ábrázolási tartományokat a MATLAB® automatikusan választotta, a tartomány választásánál igazodott a függvény értelmezési tartományához és értékkészletéhez. A fenti ábránk szemléletes ugyan, de ugyanakkor elég „kopasz”. Cicomázzuk ki egy kicsit!

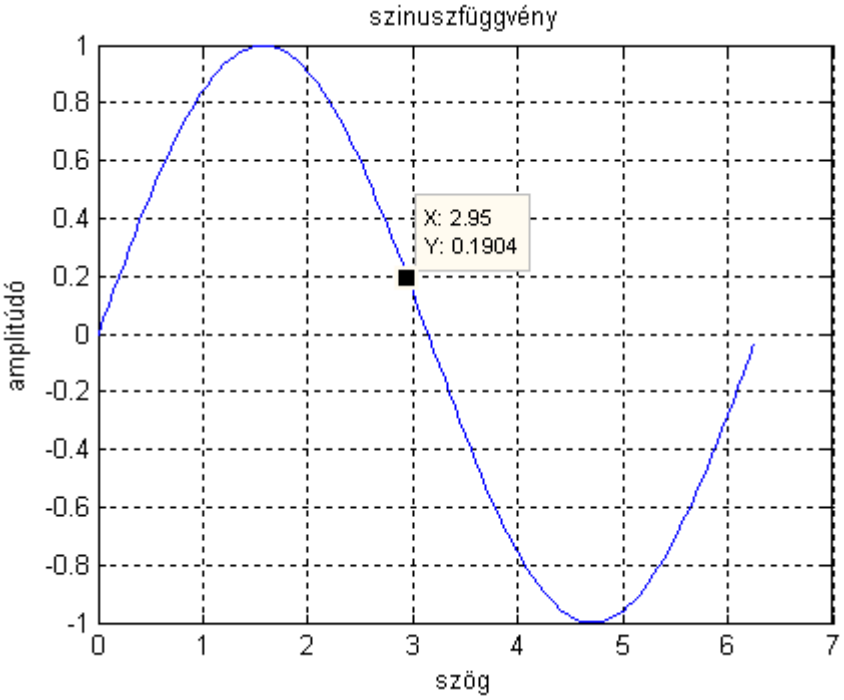
```
>> grid;
>> title('szinuszfüggvény');
>> ylabel('amplitúdó');
>> xlabel('szög');                    (4.4.)
```

A „felöltöztetett” ábrát (4.2 ábra) most már könnyebben tudjuk értelmezni. A „grid” utasítás hatására az ábrára az egyes értékek leolvasását segítő „raszter” háló kerül. A „title” függvénnyel nevet adhatunk az ábránknak, míg az „xlabel” és az „ylabel” függvény az x és az y tengely feliratozását teszi lehetővé.



4.2 ábra Szinusz függvény cicomával

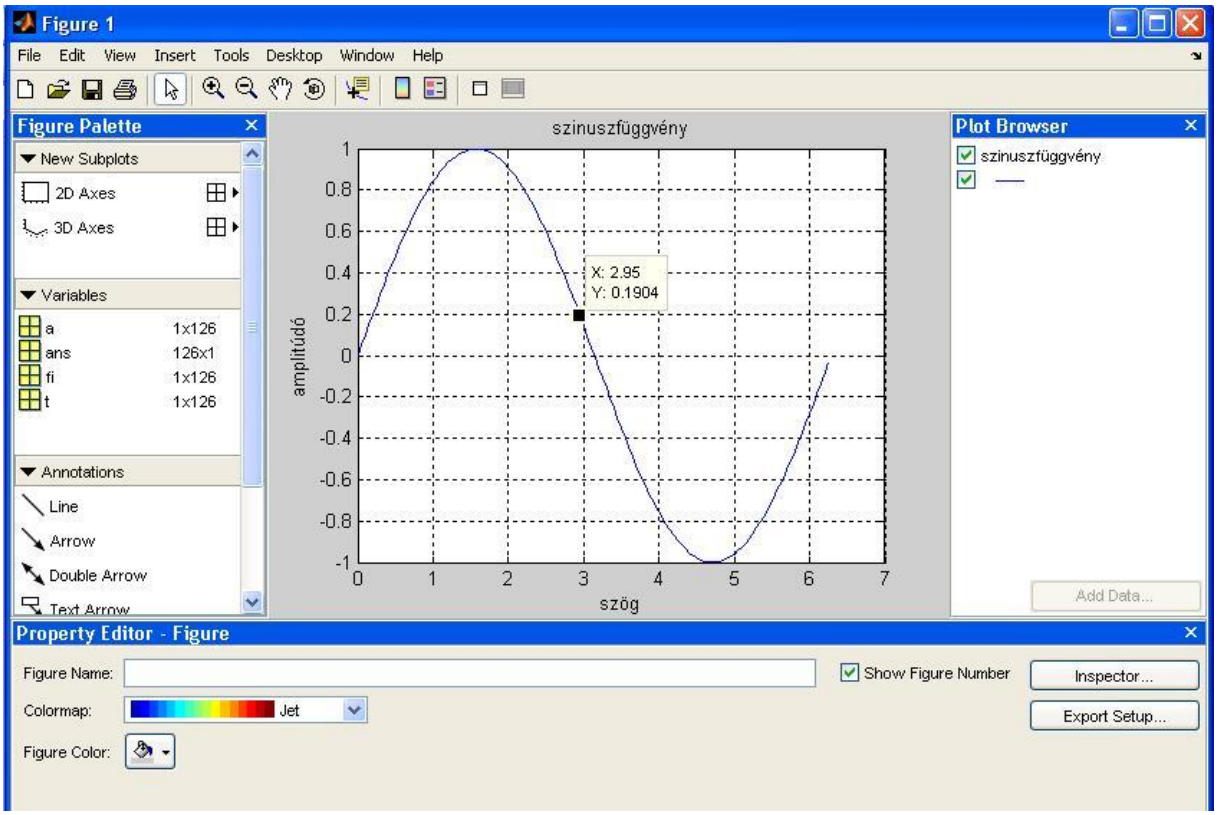
A grafikus ábrákon (4.1 és 4.2) menüpontokat láthatunk, melyek segítségével további szerkesztés valósítható meg. Egyes gyakran használt menüpont ikonként is megjelenik (ábra mentése, ábra nyomtatása, ábra- forgatás, nagyítás-kicsinyítés, adatkurzor ki- bekapcsolás, „plot-tools" ki- és bekapcsolás). Ezen menüpontokat jelen jegyzetben nem részletezzük, de néhány hasznos funkciót áttekintünk.
Dokumentáció készítéséhez nagyon hasznos funkció, hogy az ábrák (MATLAB® ablak és keret nélkül) közvetlenül fájlba menthetők. A fájl formátum paletta meglehetősen széles (bmp, jpg, pcx, pdf.. stb). A fájlba mentett ábrák rajzoló programmal (mint bármilyen más kép) szerkeszthetők, szövegszerkesztővel készített dokumentumba beilleszthetők.
Legtöbb program esetében a vágólap használatával is felhasználhatjuk ábráinkat más programokban. A 4.2. ábra szerinti ablakban kattintsunk az Edit menü Copy Figure pontjára, így a görbénk a szürke háttér nélkül máris a vágólapra került.
Formázás nélkül a 4.4 ábra kép fájlba mentés után, majd jelen dokumentumba illesztve a következőképpen mutat:



4.3 ábra Fájlba mentett kép formázás nélkül

Mint hasznos lehetőséget, érdemes megemlíteni a „data cursor” ki- be kapcsoló ikont (az ikon sorban jobbról balra az 5. ikon), melyre ha rákattintunk, majd a kurzorral a függvény bizonyos nevezetes pontjára mutatunk, egy szövegdobozban a program kiírja az adott pont x és y értékét.

A plot-tools bekapcsolása után a következőt láthatjuk a képernyőn:



4.4. ábra Bekapcsolt „plot-tools”

A plot-tools (menüpont alatti ikonsor jobb szélső eleme) bekapcsolása után az ábránk köré újabb ablakok nyílnak. A „szerszámkészlet” segítségével az ábránkat szinte korlátozás nélkül csinosíthatjuk.

Ha a megjelenített grafikonunkat a fenti eszközkészlettel kedvünkre formáztuk, felmerülhet a kérdés, hogy legközelebb ugyanezeket a beállításokat, hogyan tudnánk egyszerűen újra alkalmazni más adatok megjelenítésére. A függvény beállításait egyszerűen menthetjük el egy „m” fájlba, melyet a MatLab automatikusan képes generálni: a készre formázott grafikon ablakában kattintsunk az ábrán jobb gombbal, majd válasszuk a „Show M-code” pontot. Ekkor megjelenik az Editor ablak, melyben egy createfigure nevű függvény látható. A sorok között böngészve felfedezhetjük a függvény kinézetével kapcsolatos beállításainkat.
A függvény két bemenő paraméterrel dolgozik, az x és y értékpárokat várja paraméterként (createfigure(x1, y1))
Mentés után a függvényünk hívható a parancsablakból az elmentett fájl nevével, plusz a szükséges paraméterek megadásával:

```
createfigure(fi,a);
```

Az m kódban a függvény általunk megadott címét, a tengelyek feliratát, stb. helyettesíthetjük változó nevekkel. Ezeket soroljuk fel a függvény paraméterlistájában is, így a függvény bármilyen jellemzője megadható minden függvényhíváskor. Pl.:

```
createfigure(fi,a,figtitle,x_axe,y_axe);
```

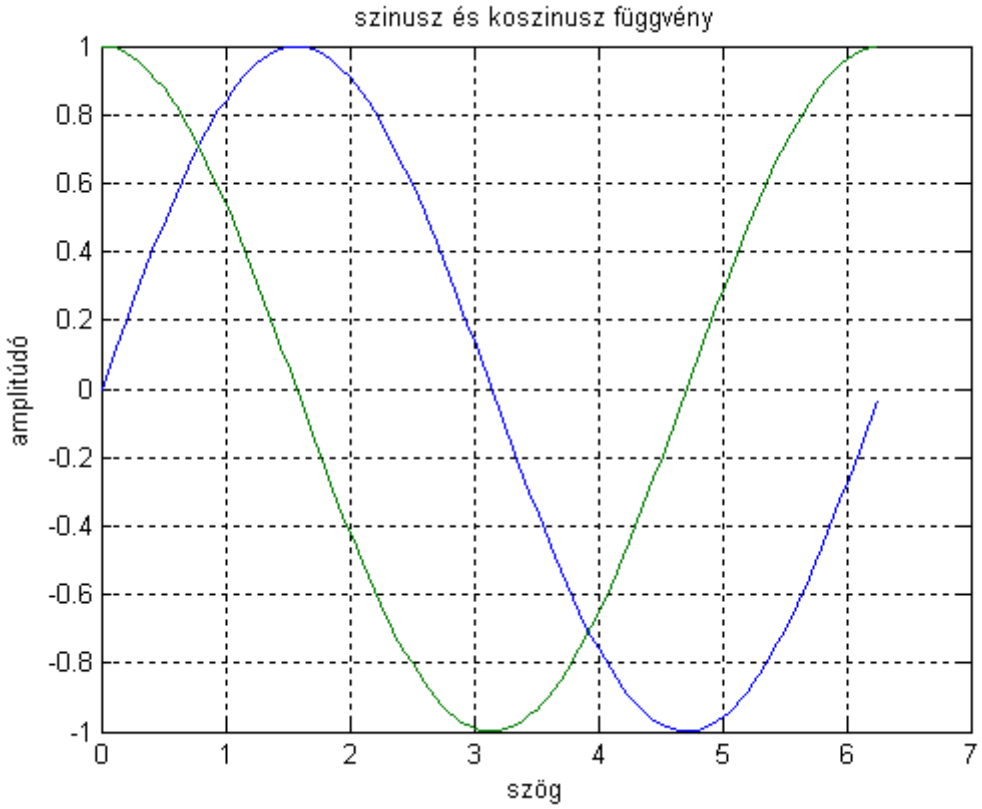
Gyakran előfordul az a feladat, hogy több függvényt kell megjelenítenünk. Erre a MATLAB® több lehetőséget is ad. A következő példában állítsunk elő egy koszinusz függvényt az előzőleg használt fi vektor (4.1.) alkalmazásával.

```
>> b=cos(fi);                                    (4.5.)
```

Jelenítsük meg a korábban előállított (4.2.) és az **a** vektorban tárolt szinusz függvény mintákat, valamint a most (4.5.) előállított és **b** vektorban tárolt koszinusz függvény mintáit egy közös grafikonban. A megjelenítéshez a következő, már az előzőekben megismert parancsokat kell gépelnünk a MATLAB® parancs ablakba:

```
>> plot(fi,a,fi,b);
>> grid;
>> title('szinusz és koszinusz függvény');
>> ylabel('amplitúdó');
>> xlabel('szög');                               (4.6.)
```

Az ábra ablakban megjelenik a szinusz és a koszinusz függvény:



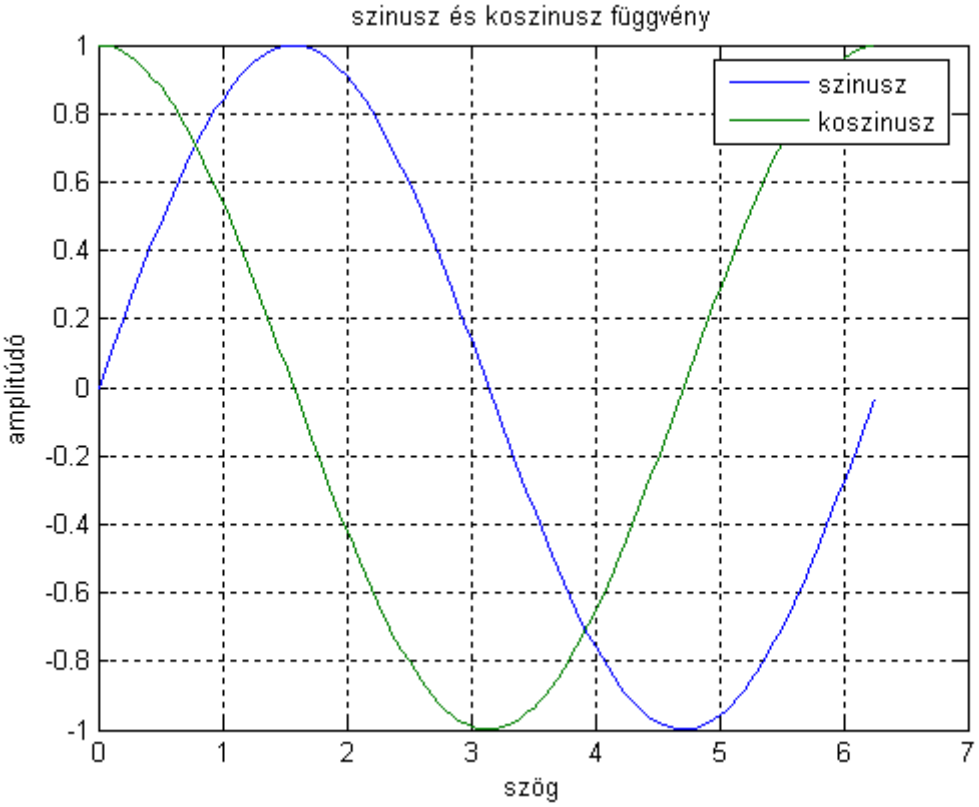
4.5 ábra Szinusz és koszinusz függvény ábrázolása közös koordináta rendszerben

A 4.5 ábrán az a és a b vektort közös koordináta rendszerben jelenítettük meg. A (4.6.) parancsok végrehajtása után a megnyíló „figure” ablak „file” menüpont „save as” pontjával fájlba mentettük a képet. A fájlnév megadás során a fájl típusa is megválasztható. A JPEG (jpg kiterjesztés) tömör formátumot ad, de ekkor az ábra rosszabb minőségű lesz, valamint jelen esetben (kontrasztos ábra) a színinformáció is jelentősen sérül. Abban az esetben, ha BMP (bitmap) formátumot választunk, akkor az előbb említett negatívumok (szín és kontraszt minőség romlás) nem jelentkezik, de ekkor nagyobb méretű fájlok keletkeznek.
A MATLAB® a közös diagramban megjelenített függvényekhez tartozó görbéket a megkülönböztethetőség érdekében különböző színnel ábrázolja. Meg kell itt említenünk, hogy a megjelenítés színe és a megjelenítés módja a felhasználó által beállítható paraméter. A paramétereket a „plot” függvénynél adhatjuk meg. A 4.5 ábrán triviális, hogy melyik görbe melyik függvényhez tartozik, de ez nem minden esetben ilyen egyértelmű, ezért lehetőségünk van kiíratni az ábrára azt, hogy az első, második stb n-edik függvényünket milyen színnel ábrázoltuk. Ezt a célt szolgálja a „legend” függvény. A legend függvény alkalmazásánál a megjelenítendő függvény nevet kell megadnunk. Az ábránkon ekkor megjelenik egy szövegdoboz. A szövegdobozban a függvényt megjelenítendő „vonal-” és szín- minta, valamint a paraméterként megadott felirat. A plot és a legend függvény között matematikai kapcsolat nincs, kapcsolat kizárólag a paraméterek sorrendjéből fakad. Írjuk be a parancs ablakba a következőt:

```
>> legend('szinusz','koszinusz');                    (4.7.)
```

A parancs (4.7.) végrehajtás után megjelenik az előbb említett szövegdoboz (lásd 4.6 ábra). Emlékezzünk vissza, hogy a szinusz és a koszinusz függvény megjelenítésekor (4.6.) a plot függvény paramétereként először a szinusz x és y paraméterét adtuk meg, tehát a legend

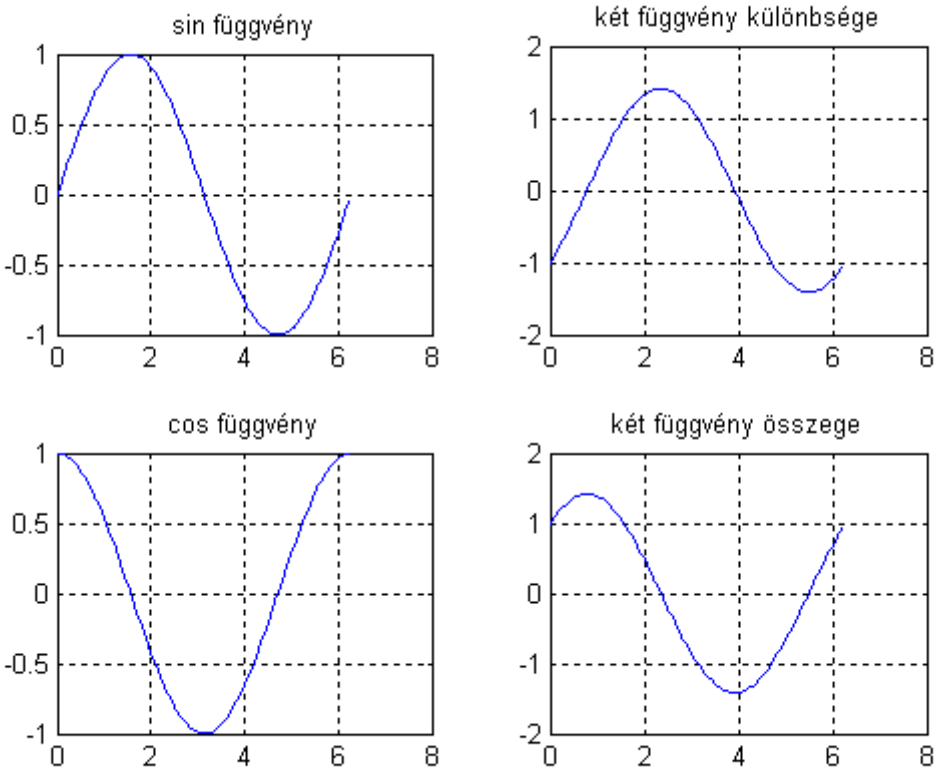
paraméterezésekor (4.7.) először a ’szinusz’ paramétert kellett megadni. Abban az esetben, ha a legend paraméterezését felcserélnénk, hibás ábrát kapnánk!



4.7 ábra Függvénynév megjelenítés a „legend” függvénnyel

Gyakran jelentkezik az az igény, hogy ábráinkat úgy jelenítsük meg, hogy azok ne legyenek egymásra vetítve, hanem azok egymás mellett (vagy egymás felett, esetleg mindkettő módon együttesen) legyenek. A MATLAB® erre is ad lehetőséget. Nézzük a következő példát!

```
>> subplot(2,2,1), plot(fi,a); grid; title('sin függvény');
>> subplot(2,2,2), plot(fi,a-b); grid; title('két függvény különbsége');
>> subplot(2,2,3), plot(fi,b); grid; title('cos függvény');
>> subplot(2,2,4), plot(fi,a+b); grid; title('két függvény összege');          (4.8.)
```



4.8 ábra Több függvény megjelenítés a subplot-tal

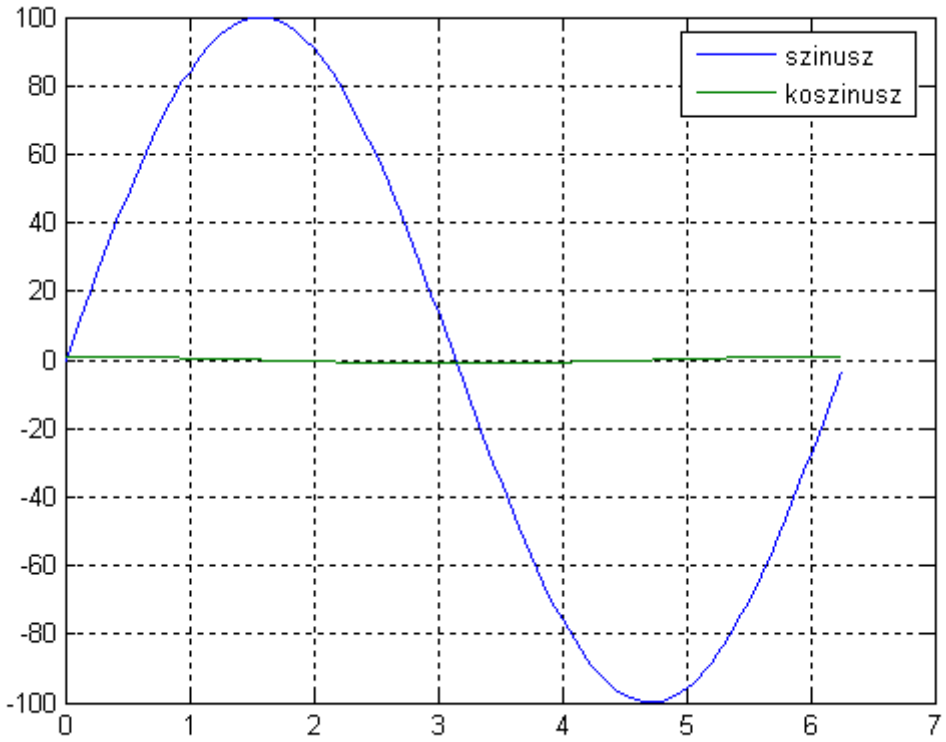
A subplot függvény lehetőséget biztosít az ábrák egyidejű, de külön-külön koordináta rendszerben történő megjelenítésére. A subplot függvény három paramétert vár. Az első paraméterként azt kell megadni, hogy egymás mellett hány ábrát kívánuk megjeleníteni, a második pedig jelzi az egymás alatti ábrák számát. Harmadik paraméterként az aktuális ablak sorszámát (mint al-ablak) kell megadni, melyben majd a már jól ismert plot paranccsal végezzük el az ábrázolást. A (4.8.) programrészletben egy sorba több parancsot írtunk, mindegyik parancsot pontosvesszővel választottuk el, kivéve a plot és a subplot-ot. Ezen parancsok most szervesen összetartoznak (subplot jelöli ki a felosztást és az aktuális rajz teret, a plot pedig elvégzi a megjelenítést), azokat vesszővel választjuk el.
A 4.8 ábrát a már előzőleg megismertek szerint tovább csinosíthatjuk (például elnevezetjük az x , valamint az y tengelyt, a megjelenítéshez színt választhatunk, kiválaszthatjuk továbbá a megjelenítés módját is).

Újabb problémával szembesülhetünk, ha két olyan függvényt kívánunk ábrázolni egy közös koordináta rendszerben, melyek amplitúdója egymástól nagy mértékben különbözik.
Futtassuk a következő programot!

```
>> fi=[0:0.05:2*pi];
>> a=100*sin(fi);
>> b=cos(fi);
>> plot(fi,a,fi,b);
>> grid
>> legend('szinusz','koszinusz');
```

(4.9.)

Ekkor a következő ábra jelenik meg:

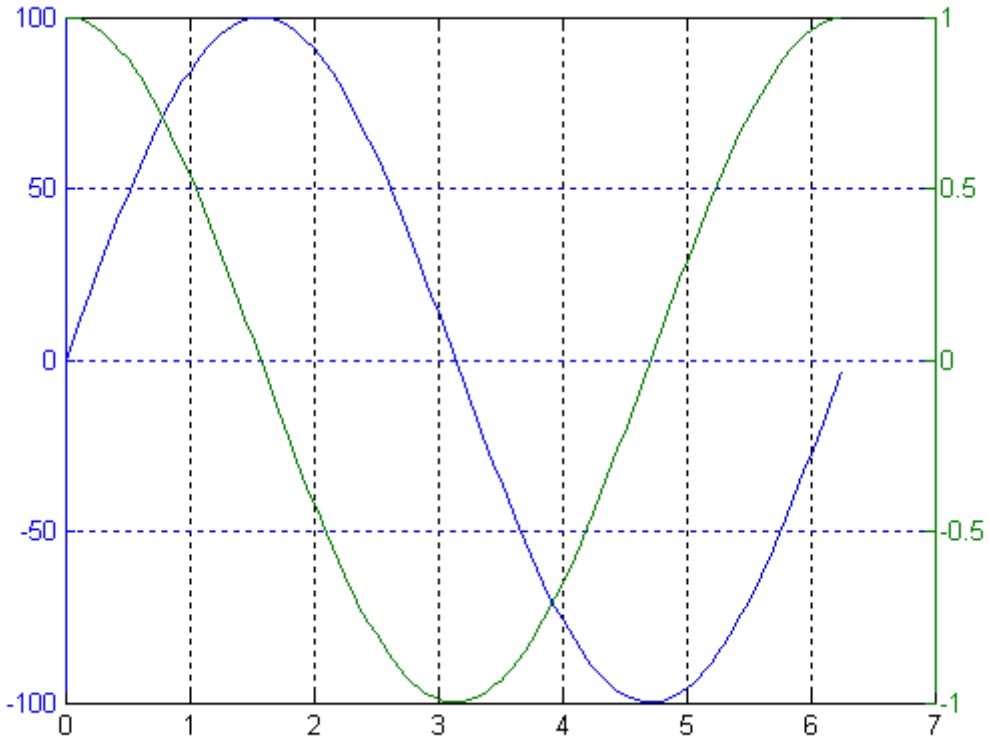


4.9 ábra Különböző amplitúdójú függvények ábrázolása közös koordináta rendszerben

A 4.9 ábrán láthatjuk, hogy a nagy mértékben eltérő amplitúdójú függvények esetén a y tengely automatikus skálázása miatt a kis amplitúdójú jel alig látszik. Természetesen a megjelenítés előtt lehetőségünk adódik megszorozni a kis amplitúdójú függvényt egy konstans számmal, de ekkor helytelenül skálázott ábrát láthatunk. Ezt az eltérést célszerűen jeleznünk kell a „legend” függvényben megadott kiírásnál.
A MATLAB® további lehetőséget biztosít a különböző amplitúdójú jelek megjelenítésére, ugyanis az y tengelyt kétféleképpen lehet skálázni, ha a megjelenítéshez a plotyy függvényt vesszük igénybe.
Futtassuk a következő programrészletet (feltételezzük, hogy az a és a b vektor, melyet a (4.9.)-ben definiáltunk még megvan):

```
>> plotyy(fi,a,fi,b);
>> grid;                                                    (4.10.)
```

A (4.10.) futtatás eredményeképpen a következő ábrát kaptuk:



4.10 ábra Eltérő amplitúdójú függvények ábrázolása azonos koordináta rendszerben különböző y tengely skálázással

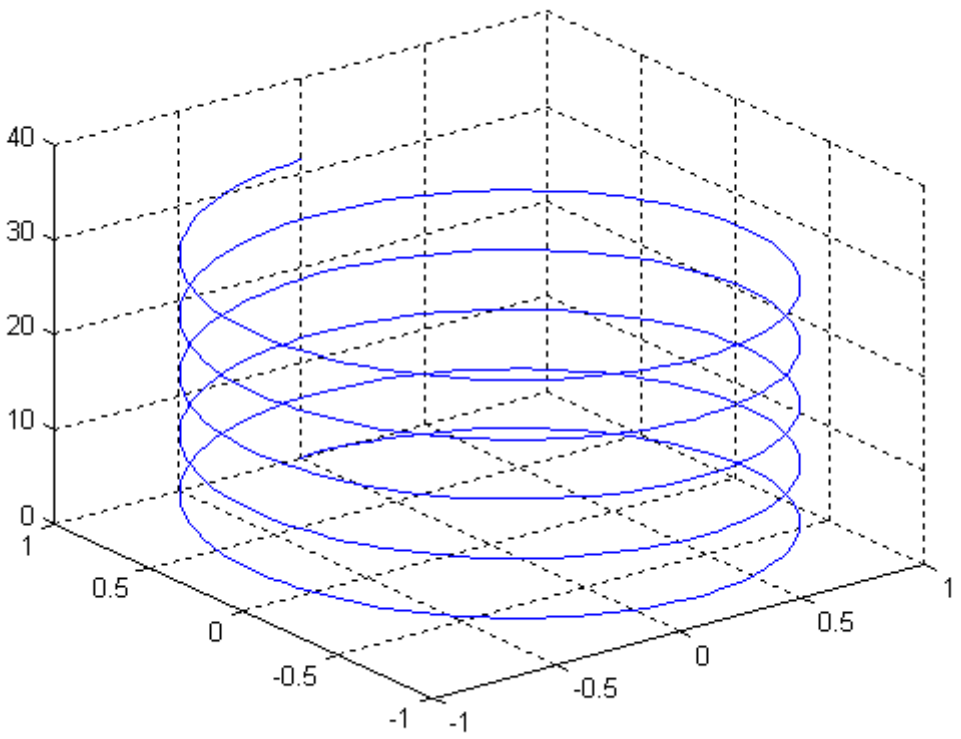
Az előzőekben megjelenített ábrák y és x tengelye lineáris osztású volt. A műszaki gyakorlatban elterjedt, hogy a tengelyek logaritmikus léptékűek (gondoljunk csak egy jel frekvenciatartományban történő analízisére, valamint egy hálózat átviteli függvényének megjelenítésére!).

Abban az esetben, ha az x tengelyt logaritmusos osztásúvá kívánjuk alakítani egy adott függvény ábrázolásakor, akkor a plot függvény helyett a megjelenítéshez használjuk a „semilogx" függvényt. Az y tengely logaritmusossá tételéhez a „semilogy" –ot alkalmazhatjuk. Ha mindkét tengely (y és x) logaritmusos skálázása szükséges, akkor ezt a „loglog" függvénnyel érhetjük el. A fent felsorolt függvények paraméterezése ugyanúgy történik, mint a „plot" függvényé.

Többváltozós függvények esetén szemléletes megjelenítési (függvény rajzolási) lehetőségünk van, melyet a háromdimenziós plot nyújt. Mindjárt nézzünk is egy példát!

```
>>  t = 0:pi/50:10*pi;
>>  plot3(sin(t),cos(t),t);
>> grid;                                                    (4.9.)
```

És az eredmény (azt hiszem elég beszédes):



4.9 ábra Háromdimenziós függvényábrázolás a plot3 függvénnyel

A plot3 függvényt a továbbiakban nem részletezzük, a pontos paraméterezése a „help plot3” parancs kiadásával listázható. A Kedves Olvasó (persze, ha van rá energiája) számtalan színes, háromdimenziós grafikában gyönyörködhet a MATLAB® demo indításával, jelen jegyzetben erre nem adunk útmutatást. A továbbiakban áttekintjük – mint a MATLAB® forrásprogramozás alapvetően szükséges elemét – a ciklusszervezést, majd a példafeladatokban most már kizárólag híradástechnika vonatkozású feladatokat oldunk meg.

Ciklusok és ciklusműveletek, feltételes elágazás

A programoknál gyakran igényként merül fel, hogy egy programrészletet többször –egy bizonyos ciklusszámnak megfelelően, vagy egy bizonyos feltétel teljesüléséig illetve nem teljesüléséig- futtassunk le. Az ilyen igények megoldására úgynevezett ciklust tudunk szervezni. Ciklusok szervezésére a MATLAB-ban is több lehetőségünk adódik.

„FOR” ciklusok

```
FOR I = 1:N,
        FOR J = 1:N,
           A(I,J) = 1/(I+J-1);
        END
    END                                                    (5.1.)
```

A fenti példában (5.1.) két egymásba ágyazott ciklust láthatunk. A külső ciklus változója az „I”, mely értéke 1 és „N” között fut. A belső ciklus változója a „J”, mely értéke szintén 1 és

„N” között vesz fel növekvő értéket. A ciklus magját egy értékadó utasítás realizálja. Természetesen a ciklus magba komolyabb, több utasítássorból álló programrész is kerülhet. N= 4 esetén a futtatás eredménye a következő lesz:

```
>> N=4;
>> for I=1:N,
for J=1:N,
A(I,J)=1/(I+J-1);
end
end
>> A

A =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

(5.2.)

Az (5.1.)-ben definiált ciklussal, N=4 érték esetén a ciklusmagba épített A(I,J)= 1/(I+J-1); értékadással egy 4 x 4-es mátrixot állítottunk elő. Az egyes elemek számításakor az aktuális I és J értékeket is felhasználtuk.
Az (5.1.) példában két egymásba ágyazott ciklust láttunk, de természetesen több ciklus is egymásba ágyazható.

Egydimenziós mátrix (vektorok) feltöltésére használhatjuk a sinus függvény rajzolásakor alkalmazott módszert is:

```
fi=[0:0.05:2*pi];
```

a szögletes zárójelben az első szám a kiinduló érték (0), ezt 0.05-tel növelve egészen 2pi-ig az összes számot elmenti a fi változóba (összesen 126 érték).

Feltételes program elágazásra ad lehetőséget a „IF” feltételes elágazó utasítás és a hozzá hasonló feltételes vezérlés átadó utasítások.

```
>>    if I = = J
        A(I,J) = 2;
      elseif abs(I-J) = = 1
        A(I,J) = -1;
      else
        A(I,J) = 0;
      end
```

(5.3.)

A fenti programrészlet az (5.2.) futtatás eredményeit felhasználja. Futtatása előtt értéket kell adnunk „I”-nek és „J”-nek. A programrészlet a következőképpen működik. Ha „I” értéke megegyezik „J” értékével, akkor az A mátrix (I,J) elemét az adott index pozícióban 2-re

cseréli. Ha az I-J abszolút értéke 1, akkor az A mátrix (I,J) eleme -1 lesz. Abban az esetben, ha egyik feltétel sem teljesül, akkor az indexelt elem értéke 0-nak adódik.
Ezt (5.3.) szavakkal a következőképpen írhatjuk le:

Ha az I egyenlő J-vel, akkor az A mátrix (I,J) eleme legyen egyenlő 2-vel;
Ha viszont I-J abszolút értéke egyenlő 1-el, akkor A mátrix (I,J) eleme legyen egyenlő -1-el;
Különben (ha a fenti két feltétel közül egyik sem teljesül), akkor A mátrix (I,J) eleme legyen egyenlő 0-val;

Az (5.3.) példa futtatása az I=4 és a J=4 kiindulási feltétellel a következő eredményt adja:

```
>> A

A =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    2.0000
```

(5.4.)

Mivel ebben az esetben az I=J feltétel teljesült, ezért az A mátrix 4. sor 4. oszlop eleme a 2 értéket kapta.

A MATLAB® ciklusszervezési és feltételes program elágazási lehetőségei természetesen sokkal tágabbak, mint azt a fentiekben megismerhettük. További példákat, magyarázatokat a „help if”, vagy „help for” parancs ablakba írásával láthatunk.

Irodalom

Richard Lyons – Quadrature Signals: Complex, But Not Complicated

Selmeczi Kálmán, Schnöller Antal – Villamosságtan II. (49 203/II.)

Ferenczy Pál – Hírközlés Elmélet (Tankönyvkiadó, Budapest 1974)

Gordos Géza – A Hírközlés rendszerelmélete (Budapest 1983 V.107)

Géher Károly, Solymosi János - Lineáris áramkörök tervezése

MATLAB® help

www.mathworks.com