Fejezet 1

Bevezetés

A computeralgebra fogalmának tisztázása érdekében a computeralgebrában előforduló számításokat a numerikus számításokkal hasonlítjuk össze. Mindegyiket egy számítógép segíti, vannak azonban alapvető különbségek, melyeket most részletesebben is bemutatunk.

1.1 Numerikus számítások

Numerikus számítások esetén egy matematikai problémát közelítőleg oldunk meg, a számítási lépések *számokkal* történnek. Ezek a számok intern *tizedes-tört-alakban* vannak tárolva, így az aritmetikai operációk gyorsan kivitelezhetőek. Ezen megjelenítésnek megvan azonban az a hátránya, hogy sem a számítások, sem az eredmények nem pontosak, hisz többek közt a kerekítések miatt is hiba lép fel. A numerikus algoritmusokat általában úgy állítják össze, hogy minél hamarabb adjanak közelítő eredményt. Gyakran a közelítés az egyetlen lehetőség egy matematikai probléma megoldására, ha nem létezik zárt alakú, pontos megoldás. Ezen kívül hasznosak még a közelítőmegoldások olyan esetekben, mikor pontos végeredményre nincs is szükségünk (pl. a vizualizálásnál).

1.2 Computeralgebra

A számokkal történő numerikus számításokkal ellentétben a computeralgebra esetén *szimbolikus* számításokat végzünk, [Hec 93] alapján *"matematikai objektumokkal való számolásokról"* van szó. Egy objektum lehet például egy szám, de egy polinóm, egy egyenlet, egy kifejezés vagy egy formula, egy függvény, egy csoport, egy győrű vagy egy másik tetszőleges matematikai objektum is. A számokkal történő szimbolikus számítások *pontosak*, szemben a numerikus számításokkal, mivel itt intern tetszőleges hosszúságú egész és racionális számok megjelenítése syerepel. Ezt a számítást a computeralgebrában *szimbolikus*, vagy másnéven *algebrai* számításnak nevezzük. [Hec

93]-ban a következő definíciót találjuk rá:

1. A "szimbolikus" jelző azt jelenti, hogy a cél a megoldás lehetőleg zárt formulájának megtalálása, jó (azaz egyszerű), szimbolikus megjelenítésben.

2. Az "algebrai" jelző a *pontos* számításra utal, ellentétben a tizedesjegy-aritmetikára jellemző közelítőmegoldással.

A computeralgebrát néha "szimbolikus manipulációnak" vagy "formulamanipulációnak" is nevezik, mivel formulákkal és szimbolumokkal történik a számolás. Példa erre a szimbolikus integrálás vagy a differenciálás, mint

$$\int x\ dx = \frac{x^2}{2}, \qquad \int_1^4 x\ dx = \frac{15}{2}, \qquad \frac{d}{dx} ln\ ln\ x = \frac{1}{x\ ln\ x}$$

vagy egyenletek szimbolikus megoldásainak kiszámítása. Erre példa az $x^4 + px^2 + 1 = 0$ x ismeretlenes, p paraméteres egyenlet, melynek megoldási halmaza:

$$\left\{ \pm \frac{\sqrt{2}\sqrt{-p - \sqrt{p^2 - 4}}}{2}, \pm \frac{\sqrt{2}\sqrt{- + \sqrt{p^2 - 4}}}{2} \right\}$$

Egy pontos megoldás szimbolikus kiszámításához szinte mindig több számolási időre és memóriára van szükség, mint a numerikus megoldás kiszámításához. A szimbolikus eredmény azonban pontos, általánosabb és legtöbbször további információt ad a problémához és annak megoldásához. Tekintsük például a fenti megoldási formulát, mely tetszőleges p paraméter esetén megadja az egyenlet megoldását: megmutatja a p-től való függőséget. Így például megtudhatjuk, hogy az eredmény hogyan változik a paraméter változása esetén.

Speciális felhasználás esetén érdemes kombinálni a szimbolikus és a numerikus módszert. Vannak a computeralgebrában például olyan algoritmusok, melyek a hardware hatékony tizedestört-aritmetikájából profitálnak. Másrészt viszont ésszerű egy numerikus problémát szimbolikusra egyszerűsíteni, mielőtt a tényleges approximatív algoritmust alkalmazzuk.

1.3 Computeralgebra-rendszerek tulajdonságai

Az ismert computeralgebra-rendszerek többsége interaktívan felhasználandó programcsomag: a felhasználó megad egy sor formulát és parancsot a rendszernek, melyet aztán a rendszer feldolgoz (vagy másnéven *kiértékel*). Végül a rendszer visszaad egy választ, melyet tovább manipulálhatunk.

A pontos szimbolikus számításokon kívül a legtöbb computeralgebra-rendszer az eredményt numerikusan is képes approximálni. Ekkor a felhasználó tetszőlegesen megadhatja, hogy hány tizedesjegy pontossággal szeretné a megoldást megkapni. `MuPAD` esetén ez a `DIGITS` globális változó segítségével történik. Így például az egyszerű `DIGITS:=100` parancssal elérhetjük, hogy a MuPAD 100 tizedesjegyre kerekítve adja meg az eredményt. Persze az ilyen számításokhoz több számolási idő és memória szükséges, mint a hardware tizedesjegy-aritmetikájának használatához.

A modern computeralgebra-rendszerek ezen kívül plusszba még egy hatalmas programnyelvvel[1] is rendelkeznek és olyan eszközöket nyújtanak, melyek lehetővé teszik a matematikai adatok vizuális megjelenítését és animációját. Sok rendszer a nyomdakész dokumentumok (úgynevezett *Notebooks* vagy *Worksheets*) előkészítését is lehetővé teszi. A MuPAD-ban is létezik egy Notebook-koncepció, mellyel azonban ezen bevezetésben nem foglalkozunk. E könyv célja, hogy bevezetésül szolgáljon a MuPAD matematikai képességeinek használatához.

1.4 Létező rendszerek

Sok computeralgebra-rendszer létezik, ezek közül néhányat kereskedelmileg forgalmaznak, míg mások ingyen hozzáférhetőek.

Az úgynevezett *special purpose* rendszerek speciális matematikai problémák kezelését segítik. Így létezik például a *Schoonship* rendszer a nagyenergiájú fizikai problémákhoz, *DELiA* a differenciálegyenletekhez, *PARI* a számelmélet[2] alkalmazásához és *GAP* a csoportelméleti problémákhoz.

Ezek mellett létezik még az úgynevezett *general purpose* computeralgebrerendszer. Ide tartozik az 1980 óta kifejlesztett és kifejezetten kisszámítógépre szánt *Derive* és *Math Plus* illetve a *Math View* (korábban *Theorist*), melyet 1990 óta fejlesztenek és mely részletesen kidolgozott felhasználói felülettel rendelkezik, aminek matematikai képességei azonban korlátozottak. Ezeken kívül létezik még a *Macsyma* és *Reduce* rendszer, melyeket 1965 óta fejlesztenek és LISP programozásúak. A modern rendszereket, mint *Mathematica* és *Maple*, körülbelül 1980 óta fejlesztik és C-ben programozták. A *Mathematica* volt az első olyan rendszer, mely felhasználóbarát felülettel rendelkezett. Meg kell még említeni az *Axiom*-ot, melyet szintén 1980-óta fejlesztenek. Az eddig említett rendszerekkel ellentétben ax Axiom egy teljes tipizált nyelvvel rendelkezik és csak speciális matematikai kontextusban engedélyezi a számításokat. Mindezen rendszerek közül a MuPAD a legfiatalabb: 1990-óta fejlesztik a Paderborni Egyetemen és megpróbálják összekapcsolni a különböző elődök erősségeit a modern, egyéni koncepciókkal.

[1] A MuPAD programnyelvének szintaxisa a Pascaléhoz hasonló. Az objektumorientált programozásnak létezik egy koncepciója.

[2] A MuPAD ezen rendszer részeit intern alkalmazza.

1.5 MuPAD

A computeralgebre-rendszer eddig említett tulajdonságai mellett a MuPAD rendelkezik még a következő képességekkel:

- A MuPAD az objektumorientált programozáshoz nyújt koncepciót. Saját adattípusokat definiálhatunk és szinte minden meglévő operátort és függvényt alkalmazhatunk rajtuk.
- A MuPAD egy interaktív forráskód-debaggart bocsájt a rendelkezésünkre.
- A modulkoncepcióval C-ben vagy C++-ban írt programokat hozzáfűzhetjük a MuPAD-maghoz.

A MuPAD szíve az úgynevezett *mag*, mely hatékonysági és gyorsasági szempontok miatt főleg C-be és részben C++-ba van implentálva. Ez a mag pedig a következő alapvető részekből áll:

- Az úgynevezett *Parser* elolvassa a rendszernek megadott adatokat és ellenőrzi szintaxisukat. A hibátlan inputot a Parser átírja MuPAD adattípusra.
- Az úgynevezett *kiértékelő* (angolul evaluator) kiértékeli az inputokat és leegyszerűsíti az eredményeket. Ez a folyamat a MuPAD-ban pontosan van definiálva és ezzel később még részletesen foglalkozunk.
- A memóriakezelő a felelős a MuPAD-objektumok intern kezeléséért.
- Egyes gyakran alkalmazott algoritmusok, mint például az aritmetikai függvények, hatékonysági és gyorsasági okokból *magfüggvényként* a C szinten vannak implentálva.

A magban van a MuPAD-programnyelv is definiálva. E nyelv segítségével vannak a MuPAD-hoz tartozó programkönyvtárak implentálva, melyek a rendszer matematikai képességeit tartalmazzák.
Emellett a MuPAD kényelmes felhasználói felülettel is rendelkezik az úgynevezett *Notebooks* vagy ábrák létrehozásakor és a MuPAD-nyelveken írt programok debuggolásakor. A MuPAD-sógúrendszernek hypertextfunkcionalitása van. Navigálhatunk a dokumentumokban, de egérkattintással példákat is kiszámíttathatunk a rendszerrel. A MuPAD-rendszer főösszetevőit az 1.1-es ábra szemlélteti.

Fejezet 2

Első lépések a MuPAD-dal

A computeralgebra-rendszereket, mint a MuPAD, gyakran interaktív módon használjuk, azaz megadjuk a rendszernek az utasításokat, mint például két szám szorzását és várunk amíg a MuPAD kiszámolja az eredményt és kiírja a képernyőre. Futó számolás közben nem adhatunk be több adatot, de a számítást bármikor megszakíthatjuk.

A MuPAD/program meghívásával elkezd futni a program. A megnyitás függ a felhasznált operációs-rendszertől és az alkalmazott MuPAD/verziótól. Itt az MuPAD installálásának leírásában szereplő információkra támaszkodhatunk. A MuPAD rendelkezésünkre bocsájt egy súgórendszert, melytől a program futása közben is kaphatunk felvilágosítást a rendszerfüggvényekről, azok szintaxisairól és egyebekről. A MuPAD-súgó használatát a következő bekezdésben ismertetjük. A súgóoldalak megnyitása lesz legalábbis eleinte a a legfontosabb rendszerparancs, melyet a kezdő alkalmaz. Ezt követi egy bekezdés, mely a MuPAD-ot, mint "Zsebszámológépet" mutatja be: a számokkal való számítást. Ez a legegyszerűbb és legintuitívabb része e leírásnak. Ezután bemutatunk néhány szimbolikus számítást szolgáló rendszerfüggvényt. Ez a fejezet nem túl szisztematikus: igazából csak egy első benyomást kíván adni a rendszer szimbolikus képességeiről.

A program elindítása után MuPAD-nyelven megadjuk a parancsot. A rendszer ekkor input-módban található, azaz vár az inputra, ha a MuPAD-prompt megjelent. Windows illetve Macintosh esetén ez a prompt a •jel, UNIX esetén >>. Ezután a példákon való illusztrálás esetén a Unix-promtot használjuk. A <RETURN>-billentyű vagy a <SHIFT>+<RETURN>-kombináció megnyomásával (Windows és UNIX esetén) befejezzük az inputot és a MuPAD kiértékeli a megadott parancsot. Macintosh esetén a parancs teljesítése érdekében az <ENTER>-billentyűt kell megnyomni, a <RETURN>-billentyű alkalmazása itt csak sorváltást eredményez és a MuPAD továbbra is input állapotban található. A MuPAD 1.4 változatáig minden parancsot pontosvesszővel vagy kettősponttal kell lezárnunk, mielőtt a <RETURN>/<ENTER>-billentyű segítségével kiértékeltetjük a maggal. Későbbi verziók ilyen lezárási

jelek nélkül is kiértékelik a helyes szintaxisú parancsokat <RETURN>/<ENTER> után.
Ha a

```
>> sin(3.141);
```

parancsot <RETURN> illetve <ENTER> követi, a képernyőn a következő eredmény jelenik meg:

```
 0.0005926535551
```

Itt a 3.141 szinuszának értékét kerestük, eredményül a közelítő szinusz-értéket tizedestört alakban kaptuk meg, ahogy azt egy zsebszámológép is kijelezte volna.
Ha a pontosvessző helyett kettősponttal zárunk le egy parancsot, akkor a MuPAD elvégzi ezt ugyan, a képernyőn azonban nem jelenik meg az eredmény. Így lényegtelen részeredményeknek nem kell megjelenniük a képernyőn. Egy sorban több parancsot is megadhatunk. Pontosvessző vagy kettőspont útján kell őket egymástól elválasztani:

```
>> diff(sin(x^2), x); int(last(1), x);

           2

  2 x cos(x )

       2

  sin(x )
```

Ekkor `x^2` x négyzetét jelöli, a diff illetve int MuPAD-függvények pedig a "differenciálás" és "integrálás" matematikai műveleteket végzik (7. fejezet). A last(1) meghívása az utolsó kifejezésre vonatkozik (most tehát $\sin(x^2)$ deriváltjára). A last-tal kapcsolatos mechanizmust a 12. fejezetben mutatjuk be.
A következő példánál csak a második parancs eredménye fog megjelenni a képernyőn, mivel a kettőspont használatával elnyomtuk a az első parancs eredményének kijelzését:

```
>> egyenletek:= {x + y = 1, x - y = 1}: solve(egyenletek)
```

Itt az `egyenletek` megnevezéshez egy két egyenletből álló halmazt rendelünk. A `solve(egyenletek)` (angol: to solve=megoldani) parancs adja az eredményt. Az egyenletmegoldó leírásáról a 8. fejezet szól.
Ha egy hiányos parancsot (ahol pl. lefelejtettük a pontosvesszőt) akarunk a <RETURN>-billentyű megnyomása által a MuPAD-dal kiértékeltetni, akkor

a képernyőn egy másfajta prompt jelenik meg. E megváltozott prompt UNIX esetén `&>` alakú, Macintosh esetén egy `*` Windows esetén pedig nem jelenik meg promtjel. Ez jelzi, hogy a parancs még nem teljes. Amint a következő sorban kiegészítettük a parancsot, pontosvesszővel vagy kettősponttal lezártuk és elküldtük, a rendszer kiértékeli:

```
>>solve (sin(x) = 1, x);

  { / PI  5 PI   3 PI     \ }
  { | --, ---, - ---, ... | }
  { \ 2    2      2       / }
```

A `quit` kulcsszó meghívásával kilépünk a MuPAD-programból:

```
>>quit
```

2.1 Magyarázatok és a súgó

Ha Ön nem tudja, hogy hogyan néz ki egy MuPAD-parancs helyes szintaxisa, akkor a szükséges információt rögtön a futó program közben beszerezheti a súgórendszertől. Az `info` függvény segítségével sok MuPAD-művelet esetén kaphatunk egy rövid angol magyarázatot:

```
>> info(solve);

  solve -- solve equations and inequations [try ?solve for options]

>> info(ln);

  ln -- the natural logarithm
```

Részletesebb információt az adott függvény súgóoldalán találunk. Ezt `help("műveletnév")` útján hívhatjuk meg. Itt a nevet idézőjel közé kell tenni, mivel a `help`-függvény a jelsorozatokat inputnak tekinti, melyeket a MuPAD " segítségével képez (fejezet 4.12). A help rövidítésére szolgál a ? operátor, mely esetén nincs szükség az idézőjelekre:

```
>> ?solve
```

Az 1.4 verziónál a súgót előhívó ? parancs egyike azon kevés kivételeknek, melyek esetén nincs szükség pontosvesszőre vagy kettőspontra a lezáráshoz[1]. A MuPAD súgóoldalai a felhasznált verziótól függően különböző képpen jelenhetnek meg. A következő példa egy ASCII-formátumú súgóoldal esetén mutatja be, hogy a ?solve parancsra milyen választ ad a MuPAD terminálverziója:

[1]Kivétel még a MuPAD elhagyását szolgáló `quit`-parancs és a MuPAD futása közben az operációs-rendszernek szánt parancsokat megadó ! szimbólum (fejezet 14.4)

```
solve -- általános megoldás

   Szintaxis:

   solve( equ , indet )
   solve( sys , unk <, MaxDegree=n> <, BackSubstitution=b> )
   solve( object )

   Paraméter:

   equ -- egyenlet vagy kifejezés
   indet -- határozatlan
   sys -- egyenletek halmaza
   unk -- határozatlanok halmaza
   n -- nemnegatív egész szám
   b -- TRUE vagy FALSE
   object -- domainelem

   Áttölthető:

   equ , sys , object

   Leírás:

   A solve( equ, indet ) megoldja az indet határozatlanú equ egyenletet és
   visszaadja a megoldások egy halmazát. ...
```

Az output többi részét most helyhiány miatt nem ismertetjük. A 2.1 ábra a megfelelő hypertext-dokumentum egy részletét mutatja, mely grafikus felület használata során jelenik meg. Erről a grafikus súgóoldalról egérkattintással juthatunk el a rendelkezésre álló függvények listájához, az indexekhez vagy a kézikönyv tartalomjegyzékéhez. A sugórendszert úgynevezett hypertext-rendszerként valósították meg. Az aktív kulcsszavak alá vannak húzva vagy be vannak keretezve. Ha rákattintunk, további magyarázatokat kapunk a kifejezéssel kapcsolatban. Mindegyik oldalon van néhány általános elágazási lehetőség, pl.előre vagy hátralapozhatunk, a következő fejezetre vagy a tartalomjegyzékre ugorhatunk. Macintosh esetén ezek a navigációs lehetőségek a súgó-menüben találhatóak. A súgóoldalon található példákat automatikusan átvihetjük a MuPAD inputablakába, ha a megfelelő, aláhúzott vagy bekeretezett promptra kattintunk.

Feladat 2.1: Nézzen utána a `diff` MuPAD-differenciálás alkalmazásmódjainak! Számítsa ki sin(x^2) ötödik deriváltját!

2.2 Számítás számokkal

A MuPAD-ot úgy használhatjuk számolások esetén, akár egy zsebszámológépet. A következő parancs egy racionális számot ad eredményül:

```
>> 1 + 5/2;
```

$7/2$

Láthatjuk, hogy a MuPAD egész és racionális számokkal való számítások esetén pontos eredményt ad (szemben a kerekített tizedestörtekkel):

```
>> (1 + (5/2*3))/(1/7 + 7/9)^2;


  67473/6728
```

A ^ szimbólum a hatványozást jelöli. A MuPAD nagyon nagy számokkal is hatékonyan tud számolni. A kiszámítandó szám nagyságának valójában csak a számítógép rendelkezésre álló memória szabhat határt. Így például az 1234-es szám 123. hatványa:

```
>> 1234^123;
```

a következő meglehetősen nagy számot adja:

170515806212727042875059727620626282654302313111068290470529619322183913834868007471366306717060598572641592314554345900570589670671499709086102539904846514793135617305563699939501046220356820273557577550700832384444147778396026387067604268570040400328704248063968069686558786501669938388338883198045915994284537241460180942971772610762859524340680101441859766279838067203567299104

Az alaparitmetikán kívül még egy sor MuPAD-függvény áll a rendelkezésünkre, melyekkel számításokat végezhetünk. Egy egyszerű példa az n természetes szám faktorizációja $n! = 1 \cdot 2 \cdot \ldots \cdot n$, amit egyszerűen a matematikai jelölésével lekérhetünk:

```
>>100!;
```

93326215443944152681699238856266700490715968264381621468592963895217599932299156089414639761565182862536979208272237582511852109168640000000000000000000000000

Az `isprime` segítségével nézhetjük meg egy természetes szám esetén, hogy prímszám-e: ez a művelet végeredményként vagy TRUE-t (igaz) vagy FALSE-t (hamis) ír ki. Az `ifactor` (angol: integer factorization) segítségével megkapjuk az adott szám prímtényezős alakját:

```
>> isprime(123456789);

  FALSE

>> ifactor(123456789);

  [1, 3, 2, 3607, 1, 3803, 1]
```

Az ifactor által visszaadott lista értelmezéséhez lásd a megfelelő súgóoldalt: ?ifactor. $123456789 = +1 \cdot 3^2 \cdot 3607^1 \cdot 3803^1$. Ennél szugesztívebb formájú eredményt ad a `Factor` művelet, mely éppúgy faktorizálja a szimbolikus kifejezéseket, mint a számokat:

```
>>Factor(123456789);

   2
  3  3607 3803
```

Nézzük most a $\sqrt{56}$ "kiszámításának" esetét! Itt szembentaláljuk magunkat a problémával, hogy ez a szám irracionális volta miatt nem állítható elő pontosan törtek segítségével. Így egy "számításnak" csak egy *lehetőség szerint egyszerű*, pontos megjelenítés lehet a célja.

2.2.1 Pontos számítások

Megpróbáljuk példák útján elmagyarázni az előbb felvetett problémát,a pontos megjelenítés nehézségét. Ha a $\sqrt{56}$-t az `sqrt` (angol: square root=négyzetgyök) parancssal adjuk meg a rendszernek, akkor a MuPAD a következő eredményt szolgáltatja:

```
>> sqrt(56);

      1/2
  2 14
```

Az eredmény a $\sqrt{56}$-t a pontos $2 \cdot 14^{1/2}$ értékre egyszerűsíti, ahol MuPAD esetén $14^{1/2}$ az $x^2 = 14$ egyenlet pozitív megoldásait jelenti. Valóban ez az eredmény legegyszerűbb pontos megadása. Vegyük figyelembe, hogy a MuPAD egy olyan objektumnak tekinti a $14^{1/2}$ kifejezést, mely rendelkezik bizonyos tulajdonságokkal (lényegében azzal, hogy a 14 négyzete egyszerüsíthető).Ilyen szimbólumokkal való számolás esetén ez automatikusan végbemegy. Például:

```
>> sqrt(14)^4;

  196
```

További példaként szolgáljon a következő határérték pontos kiszámítása:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

A `limit` függvény határértéket számol, az `infinity` pedig a MuPAD-ban a "végtelent" jelöli:

```
>> limit((1+1/n)^n,n=infiniti);

  exp(1)
```

Itt az `exp` a hatványozásfüggvényt[2] jelöli. Az e számot az `exp(1)` szimbólum jeleníti meg pontosan, ahol a MuPAD rendelkezik az ezen objektumhoz szükséges számítási szabályokkal. Így például a természetes alapú ln logaritmusra a következőt kapjuk:

```
>> ln(exp(1)^2)

  2
```

További pontos számításokkal még e bevezetés során is fogunk találkozni.

2.2.2 Numerikus közelítések

A pontos számolások mellett a MuPAD lehetővé teszi a numerikus közelítéssel való számolást is. Ha például Ön a $\sqrt{56}$ tizedestörtalakjának közelítő értékét szeretné megkapni, akkor használja a `float` (angol: floating point number=tizedestört) függvényt! Ez a függvény kiszámítja az argumentum értékének úgynevezett *tizedestörtalakját*:

```
>> float(sqrt(56));

  7.483314773
```

A közelítés pontossága a DIGITS globális változó értékétől, a numerikus számolások tizedesjegy-számától függ. Az előre beállított standardérték a 10:

```
>> DIGITS;float(67473/6728);

  10

  10.02868608
```

[2] Az E szimbólum ugyanezt az objektumot jelöli a MuPAD-ban, azaz a természetes alapú logaritmus alapját (E = exp (1) = 2.71828..).

A DIGITS-hez hasonló globális változók befolyásolják a MuPAD viselkedését, ezeket környezetváltozóknak is nevezzük[3]. A MuPAD 1.4 verziójába implentállt összes környezetváltozó listája megtalálható a MuPAD-rövidreferencia [Oev 98] "Környezetváltozók" című fejezetében. A DIGITS változó bármely tetszőleges, 1 és $2^{32}-1$ közötti egész értéket felvehet:

```
>> DIGITS:= 100: float(67473/6728); DIGITS:= 10:
```

10.02868608799048751486325802615933412604042806183115338882282996432818
737217598097502972651605231 86

A következő számítások előtt a DIGITS értékét visszavettük 10-re. Ezt az `unassign(DIGITS)` parancs segítségével is elvégezhetjük. Ha a számok *egyikét* tizedestörtlakban adjuk meg, akkor a MuPAD mindig autómatikusan közelítő értéket számol számokkal való aritmetikai operációk esetén:

```
>> (1.0 + (5/2*3))/(1/7 + 7/9)^2;

  10.02868608
```

Figyeljünk azonban arra, hogy a következő két parancsra

```
>>2/3*sin(2), 0.6666666666 sin(2);
```

egyik esetben *sem* kapjuk meg a sin(2) közelítő értékét, mivel sin(2) nem egy szám, hanem egy kifejezés, mely a sin(2) (pontos) értékét jelöli:

```
  2 sin(2)
  --------, 0.6666666666 sin(2)
     3
```

A két érték egymástól vesszővel történő elválasztása egy speciális adattípust, úgynevezett *sor*-t eredményez, melyről részletesebben a 4.6 fejezetben írunk. Ahhoz, hogy a fenti esetben tizedestörtalakú eredményt kapjunk, ismét a `float` függvényt kell alkalmaznunk[4]:

[3] Különösen figyelnünk kell, ha ugyanezt a számítást különböző DIGITS értékekkel végezzük. A MuPAD-ban egyes komplexebb numerikus algoritmusok a "remember" opcióval vannak implentálva, aminek következtében ezen algoritmusok korábbi eredményekből származnak (fejezet 18.8). Ez pontatlan numerikus eredményeket okozhat, melyeket kevésbé pontos számolás útján kaptunk. Kérdéses esetben a DIGITS meghívása előtt érdemes újrainstallálni a MuPAD-programot a `reset()` segítségével (fejezet 14.3), amivel kitöröljük a MuPAD memóriáját

[4] Figyelje meg a két utolsó számjegyet. A második parancs egy kicsit pontatlan eredményt ad, mivel a 0.666... már eleve közelítése a 2/3-nak és ez a hiba a végeredményre is kihat.

```
>> float(2/3*sin(2)), 0.6666666666*float(sin(2));

  0.6061982845, 0.6061982844
```

A legtöbb MuPAD-függvény, mint ahogy az `sqrt`, a trigonometrikus függvények, a hatványozás vagy a logaritmus is, automatikusan numerikus eredményt ad, ha az argumentuma tizedestörtalakú szám:

```
>> sqrt(56.0), sin(3.14);

  7.483314773, 0.001592652916
```

A π és e konstansokkal, melyeket PI illetve E = exp(1) jelenít meg, MuPAD-ban *pontosan* számolhatunk:

```
>> cos(PI), ln(E);

  -1, 1
```

Ha szeretnénk, a `float` művelettel itt is megkaphatjuk a konstansok egy numerikus approximációját:

```
>> DIGITS:= 100: float(PI); float(E); unassign(DIGITS):


  3.1415926535897932384626433832795028841971693993751058209749445923078164
  062862089986280348253421170 68


  2.7182818284590452353602874713526624977572470936999595749669676277240766
  303535475945713821785251664 27
```

Feladat 2.2: Adja meg a $\sqrt{27}$-4$\sqrt{3}$ és a $\cos(\pi/8)$ pontos értékét! Számítsa ki öt tizedesjegy pontossággal a numerikus közelítéseket!

2.2.3 Komplex számok

Az imaginárius $\sqrt{-1}$ imaginárius egységet a MuPAD-ban az I szimbolizálja:

```
>> I^2;

  -1
```

MuPAD esetén is a szokásos módon, x+yI alakban adhatjuk meg a komplex számokat, ahol a valós illetve az imaginárius rész (x illetve y) egészszámok vagy racionális számok vagy tizedestörtek is lehetnek:

```
>> (1 + 2*I)*(4 + I), (1/2 + I)*(0.1 + I/2)^ 3, (1 + I)/(1 - I);

  2 + 9 I, 0.073 - 0.129 I, I
```

A számolási operációk eredményét nem mindig valós és imaginárius részekre bontott alakban kapjuk meg, ha a számértékek mellett `sqrt(2)`-hez hasonló szimbolikus kifejezéseket alkalmazunk:

```
>> 1/(sqrt(2) + I);

      1

  --------

   1/2

  2    + I
```

A `rectform` (angol: rectangular form) függvény azonban a valós és imaginárius részekre bontást eredményezi:

```
>> rectform(1/(sqrt(2) + I));

   1/2

  2

  ---- + (-1/3) I

   3
```

A Re illetve Im függvények segítségével megkaphatjuk egy x+yI komplex szám valós x illetve imaginárius y részét. A komplex szám x-yI konjugáltját a `conjugate` függvénnyel számíthatjuk ki. Az abszolútértéket ($|x+yI| = \sqrt{x^2+y^2}$) az `abs` MuPAD-függvény adja:

```
>> Re(1/(sqrt(2) + I)), Im(1/(sqrt(2) + I)), abs(1/(sqrt(2) + I));

   1/2         1/2

  2           3

  ----, -1/3, ----
```

```
   3           3

>>conjugate(1/(sqrt(2) + I)), rectform(conjugate(1/(sqrt(2) + I)));

             1/2

     1      2
  --------, ---- + (1/3) I
   1/2       3
  2    - I
```