Komputeralgebra a tudományokban és a gyakorlatban

Kovács Attila*

Absztrakt

A komputeralgebra rendszerek, mint szimbolikus és numerikus számítások elvégzésére egyaránt alkalmas számítógépes rendszerek kiemelkedő jelentőségűek napjaink tudományos életében. Hatékony segítséget nyújtanak a legkülönfélébb tudományterületek művelői számára, mérnökök és ipari fejlesztések nélkülözhetetlen eszközei. A tanulmány ezen rendszerek kialakulásával, fajtáival, elméleti hátterével és alkalmazásaival foglalkozik, konkrét példákkal támasztva alá az elmondottakat. Fejlődésük lehetséges perspektíváinak elemzése mellett irodalmi összefoglaló segíti a jobb áttekinthetőséget.

1. Bevezető

"The impact on mathematics of computer algebra and other forms of symbolic computing will be even larger than the impact of numeric computing has been."[1]

Az emberiség több évezredes történelme során a társadalom fejlődése mindig szorosan kapcsolódott a tudományokéhoz. A mai értelemben vett természettudományok pedig a matematika mindent átható erejével és fantáziájával karöltve bontakoztatták ki igazán önmagukat. Az utóbbi évtizedekben a matematika egy ma már önálló tudományt, az informatikát hívta segítségül az eddig a puszta képzelőerő által áthághatatlan akadályok leküzdésére, önmaga és a többi tudományterület közötti távolság további csökkentésére. Szimbolikus, numerikus és grafikus számítások elvégzésére egyaránt alkalmas számítógépes rendszerek jelentek meg, amelyek az informatika fejlődésével, az algoritmikus gondolkodásmód térhódításával egyre jobbak és jobbak lettek. Új tudományterület született, amit komputeralgebrának neveztek el.

Hangsúlyozni kell a különbséget a numerikus és a szimbolikus-algebrai számítások között. Ámbár a *numerikus* számítások nemcsak elemi aritmetikai műveleteket (összeadás, kivonás, szorzás, osztás) foglalnak magukba, hanem olyan bonyolult műveleteket is, mint matematikai függvények numerikus értékének,

*A tanulmány a Magyar Állami Eötvös Ösztöndíj és az FKFP–0144 támogatásával készült.

[1] A.M. Cohen, et al. (ed.), *Computer Algebra for Industry: Problem Solving in Practice*, John Wiley & Sons, 1993.

polinomok gyökeinek vagy mátrixok sajátértékének meghatározása, ezek a műveletek alapvetően számokra és *csak számokra* vannak értelmezve. Sőt, ezek a számok a legtöbb esetben nem pontosak, pontosságuk az adott számítógépes architektúra lebegőpontos ábrázolási módjától függ. A numerikustól eltérően a *szimbolikus és algebrai* számítások matematikai objektumokat jelentő *szimbólumokon* értelmezettek. Ezek az objektumok lehetnek számok, mint pl. egészek, racionális számok, valós és komplex számok, algebrai számok, de lehetnek polinomok, racionális és trigonometrikus függvények, egyenletek, egyenletrendszerek, sőt olyan absztrakt struktúrák is, mint csoportok, gyűrűk, ideálok, algebrák vagy ezek elemei. A *szimbolikus* szó arra utal, hogy a probléma megoldása során a részeredmények lebegőpontos számok helyett szimbólumok, így a végeredmény is szimbolikus kifejezésben, zárt alakban keresendő. Az *algebrai* szó pedig a szimbolikus objektumokkal végzett műveletek algebrai eredetét jelenti. A komputeralgebra-rendszerek mint *számítógépes programok* alapfunkciója tehát az, hogy kielégítsék az alábbi követelményeket:

- matematikai objektumok szimbolikus ábrázolása,z
- aritmetika ezekkel az objektumokkal.

A komputeralgebra mint *tudományterület* feladata pedig erre az aritmetikára épülő hatékony algoritmusok keresése, analízise és implementálása tudományos kutatásokhoz és alkalmazásokhoz.

2. A komputeralgebra-rendszerekről általában

Sem a szimbolikus számítások igénye sem a megvalósíthatóság ötlete nem újkeletű. Arról a tényről, hogy a számok nemcsak numerikus mennyiségekként, hanem szimbólumokként is felfoghatók, így írt Lord Byron lánya, Lady Ada Augusta, Countess Lovelace 1842-ben:

"Many persons ... imagine that the business of the engine [Babbage engine] is to give results in numerical notation, the nature of its processes must consequently be arithmetical and numerical rather than algebraical and analytical. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were letters or other general symbols; and in fact it might bring out its result in algebraical notation were provisions made accordingly."

Charles Delaunay 1847-ben kezdte el számításait, amelyekben minden addiginál pontosabban megadta a Hold helyzetét az idő függvényében. Hogy a Hold aktuális pozíciójának méréséből adódó hiba halmozódását elkerülje, szimbolikusan számolt. Módszerét siker koronázta, a végső képlet 128 oldalt tett ki. A dolognak csupán két szépséghibája volt. Az egyik, hogy a számolás és ellenőrzés 20 évbe telt, a másik, hogy a számításokba 3 hiba csúszott. Ez utóbbit csak 1970-ben fedezték fel a Boeing Scientific Research Laboratories munkatársai az egyik saját fejlesztésű szimbolikusan számoló szoftver segítségével: a futási idő 20 órát vett igénybe.

A komputeralgebra-rendszerek fejlődésének története természetes módon szoros kapcsolatban áll az informatika fejlődésével. A hatvanas évek elején listakezelés céljaira tervezték a LISP programozási nyelvet, amely más nyelvektől eltérően (FORTRAN, ALGOL60) felépítésénél fogva különösen alkalmas volt szimbólumokkal történő műveletekhez. Ámbár a későbbi rendszerek közül többet már C-ben írtak, a LISP hatása és szerepe a komputeralgebrában napjainkban is rendkívül jelentős. A számítástechnika kezdeti időszakában a különböző tudományterületek művelői szimbolikus számításaik megkönnyítése és felgyorsítása érdekében kezdték el fejleszteni az első komputeralgebra rendszereket, amelyek átdolgozva, megújulva és a sokadik változatban napjainkban is jelen vannak. A legismertebb ilyen ún. *speciális célú* komputeralgebra rendszerek:

- algebrai geometria: MACAULAY ([27, 59])
- relativitáselmélet: SHEEP, STENSOR ([19, 35, 45])
- csoportelmélet: CAYLEY, GAP ([7, 8, 57])
- égi mechanika: CAMAL ([3])
- kommutatív algebra: CoCoA, MACAULAY ([9, 23, 27, 59])
- Lie-algebrák: LIE ([44])
- nagyenergiájú fizika: FORM, SCHOONSHIP ([54, 63, 61])
- számelmélet: PARI, SIMATH, KANT ([4, 25, 34])

A hetvenes években jelentek meg az *általános célú* komputeralgebra-rendszerek, amelyeket beépített adatstruktúrák, matematikai függvények és algoritmusok széles választéka jellemez, minél nagyobb felhasználói területet próbálva meg lefedni. A jelentős számítógépes erőforrásigény miatt robbanásszerű elterjedésük a nyolcvanas évek elejére, a mikroprocesszor alapú munkaállomások megjelenésének idejére tehető. A jobb hardver környezet, a hatékonyabb erőforrásgazdálkodás, rendszerfüggetlen magasszintű alapnyelv (C) használata és nem utolsósorban a társadalmi-gazdasági igények fokozatosan piaci termékké érlelték az általános célú komputeralgebra-rendszereket, ami viszont erős javulást hozott a felhasználói felület és dokumentumkészítés terén. A jól megtervezett grafikus interfész jelentőségének felismerése — ami elsőként talán S. Wolfram Notebook-jában öltött testet — lényeges szerepet játszott elterjedésükben. Az alábbiakban azokat az általános célú rendszereket soroljuk fel, amelyek vagy a legtöbbet kínálják a létező komputeralgebra algoritmusok közül vagy valamilyen különleges tulajdonságuk miatt érdeklődésre tarthatnak igényt.

- A MACSYMA ([47]) az egyik legrégebbi általános célú komputeralgebra-rendszer. Matematikai algoritmusok széles skáláját tartalmazza, de erőforrásigénye nagy, és csak kisszámú platformon futtatható. Jelen pillanatban több változata is elérhető.

- A REDUCE ([30, 46]) nagyenergiájú fizikához készített speciális célú rendszerből fejlődött általános célú rendszerré. A MACSYMA-val összevetve a lehetőségei szűkebbek. Nagy előnye viszont, hogy a forráskód elérhető, miáltal könnyen módosítható és bővíthető. A REDUCE-t aktívan fejlesztik.

- A DERIVE ([60]) az egyetlen olyan általános célú rendszer, amelyet korlátozott erőforrásokkal rendelkező számítógépekre (főleg személyi számítógépekre) terveztek. A DERIVE nem túl széles programnyelvi, de annál szerteágazóbb, a felhasználót támogató rutinkészlettel rendelkezik. Említésre érdemes, hogy a Texas Instrument TI-92 kalkulátorába DERIVE függvényeket épített.

- MuPAD ([20, 21]) — Multi Processing Algebra Data Tool. A rendszer legjelentősebb tulajdonsága a párhuzamos szerkezet. A szimbolikus és numerikus számítások párhuzamos elvégzésén túl vizualizációs lehetőségeket is magában foglal. A szoftver szabadon terjeszthető és az aktív fejlesztésnek köszönhetően matematikai tudása gyorsan nő.

- A MAPLE ([12, 13, 14, 32, 38, 52]) nyelvet a többfelhasználós működés céljainak megfelelően tervezték. Egy kis tárhelyet igénylő program (az ún. kernel) ügyel a hardverközeli feladatokra (parancsértelmezés, aritmetikai műveletek, memóriakezelés stb.) és tartalmazza a legfontosabb eljárásokat. A matematikai tudás többi része — amit szintén MAPLE-ben írtak — csak akkor töltődik be az operatív memóriába, ha erre igény lép fel. Rugalmasságának köszönhetően az eljárásokat a felhasználók könnyen beilleszthetik vagy módosíthatják. Jelen pillanatban a MAPLE az egyik legelterjedtebb általános célú komputeralgebra rendszer.

- A MATHEMATICA ([48, 65, 26, 58]) az első olyan matematikai számításokra alkalmas rendszer, amely felhasználóbarát módon egyesíti magában a szimbolikus, numerikus és grafikus lehetőségeket. Jól struktúrált felhasználói szintű programozási lehetőséggel rendelkezik és a legkülönfélébb számítógépes architektúrákon is futtatható.

- Az AXIOM ([36]) elődje a "Scratchpad", az IBM Research Center fejlesztése. Az AXIOM már az informatika legújabb eredményein alapszik: erősen típusos programnyelv, könyvtárrendszere absztrakt adattípusok hierarchikusan szervezett kollekciója, ezáltal a felhasználó az algebrai struktúrák legkülönbözőbb típusait kezelheti és definiálhatja. Az AXIOM tervezése (ami olyan objektumorientált elemeket is tartalmaz, mint adatabsztrakció, öröklődés, polimorfizmus stb.) kísérlet az elmúlt 100 év "konstruktív matematikájának" modellezésére.

Az alábbi tulajdonságokat a legtöbb általános célú komputeralgebra rendszer magában foglalja:

- interaktivitás,

- matematikai tények ismerete,

- matematikai objektumok kezelésére képes deklaratív[2], magas szintű programozási nyelv funkcionális programozási lehetőségekkel,
- az operációs rendszer és más programok felé való kiterjeszthetőség,
- szimbolikus és numerikus számítások integrálása,
- automatikus (optimalizált) C és Fortran kódszegmens generálása,
- grafikus felhasználói környezet,
- 2 és 3 dimenziós grafika, animáció,
- szövegszerkesztési lehetőség és automatikus LaTeX konverzió,
- on-line súgó rendszer.

A komputeralgebra-rendszereket *matematikai szakértői rendszerek*nek is nevezik. Napjainkban az általános célú komputeralgebra-rendszerek szédítő iramú fejlődésének lehetünk szemtanúi, elsősorban tudásuknak és széleskörű alkalmazhatóságuknak köszönhetően. Mégis hiba volna alábecsülni a speciális rendszereket, amelyek egyrészt igen fontos szerepet játszanak sok tudományterületen, másrészt sok esetben könnyebben kezelhetők és hatékonyabbak, jelölésrendszerük és algoritmusaik alacsony szintű programnyelvi implementációja miatt. Nagyon lényeges, hogy egy adott probléma megoldásához az arra legalkalmasabb komputeralgebra-rendszert válasszuk ki.

3. Komputeralgebra algoritmusok

Első matematikus:	*...szóval az algoritmusod három részből áll. Az első az input, a harmadik az output. És mi a második?*
Második matematikus:	*A második lépés? Ahol a csoda tötrénik.*
Első matematikus:	*Értem. Tudnál erről egy kicsit többet is mondani?*

A könnyebb megértés kedvéért tekintsünk pár nagyon egyszerű példát[3]:

- A több mint ezer jegyű $450!$ kiszámolása, a 41 jegyű $35!+1$ faktorizálása és az $f(x) = \sin(2\,x^3 + e^{\sqrt{1+7\,x^2}})$ függvény MacLaurin sora első ötven együtthatójának meghatározása együtt körülbelül három másodperc.
- Az $x+y+z=3, x^2+y^2+z^2=9, x^3+y^3+z^3=24$ egyenleteket kielégítő valós x, y, z értékekre az $x^4+y^4+z^4$ kifejezés kiszámolása körülbelül egy másodperc.
- Az $\int \sqrt{x}\sin(x)\cos(x)dx$ integrál zárt alakjának meghatározása kevesebb, mint két másodperc.

[2] A deklaratív programozási nyelvek a kívánt eredményt specifikálják és nem a hozzájuk vezető utat, mint ahogy az imperatív nyelvek teszik.

[3] MapleV Release 4.0b, Pentium75

Az első pontban igazi meglepetés nincs. Az egészek faktorizálására régóta ismeretesek — sajnos nem polinomiális — módszerek (Pollard, Lenstra, Morrison-Brillhart stb.), legfeljebb a rutinmunka sebessége lephet meg bennünket. De lássuk, mi a helyzet a második és a harmadik ponttal.

Ahogy a mai modern komputeralgebra-rendszerek nem létezhetnének a számítástechnikai ipar igen látványos eredményei nélkül, ugyanez elmondható a számítástudományról is. A szimbolikus objektumok reprezentációs lehetőségei és a velük való műveletek vizsgálata már a hetvenes évek elején megkezdődött, különböző realizációk is születtek. A számítástechnika fejlődése aztán új lendületet adott az algoritmusfejlesztéshez, az alkalmazások pedig újabb és újabb fejlesztést tettek szükségessé.

A természettudományok többsége jelenségeit, gondolatait matematikai egyenletekbe foglalja. A lineáris egyenletek, egyenletrendszerek szimbolikus megoldásainak vizsgálatai az ismert eliminációs módszereken alapszanak. Magasabb fokú egyenletek helyett nemritkán lineáris approximációt használunk a nemlineáris egyenletek nehéz kezelhetősége miatt. A hatvanas évek közepén Buchberger Ph.D. dolgozatában kidolgozott egy módszert tetszőleges fokú, többváltozós polinomegyenletek hatékony kezelésére, amit ma *Gröbner-bázis elmélet* néven ismerünk. Az alapgondolat a következő: tekintsük az $f, f_1, f_2, \ldots, f_r$ valamilyen test feletti n változós polinomgyűrű elemeit és döntsük el, hogy f eleme-e az $f_1, f_2, \ldots, f_r$ polinomok által generált ideálnak. Buchberger munkájára csak tíz évvel felfedezése után kezdtek felfigyelni, azóta a terület a komputeralgebra egyik legnépszerűbb ága. (A fejezet elején adott második példa megoldásához ezt a módszert használtuk.) Egy másik lehetőség nemlineáris egyenletrendszerek szimbolikus megoldására a rezultánsmódszer, ami lényegileg eliminációs lépések sorozata. A Gröbner-bázis használatának előnye, hogy segítségével a megoldáson kívül betekintést nyerhetünk a probléma struktúrájába, a rezultánsmódszeré az alacsonyabb komplexitás. A hetvenes évek közepén Collins mutatott egy másik eljárást nemlineáris egyenletek és egyenlőtlenségek egzakt valós gyökeinek meghatározására.

Lényeges előrelépés történt a szimbolikus integrálás területén is. Habár a probléma formális természete már régóta ismeretes (Liouville's Principle), csak 1969-ben tudott R. Risch hatékony algoritmust adni annak eldöntésére, hogy ha adott egy valós elemi f függvény, akkor az $\int f(x)dx$ integrál is elemi-e és ha igen, az algoritmus meg is adja azt (elemi függvények halmazán azt a halmazt értjük, amelyet $\mathbb{Q}(x)$-ból kiindulva rekurzív módon összegeket, szorzatokat, hányadosokat, logaritmikus, exponenciális és algebrai függvényeket valamint ezek kompozícióit képezve kapunk). Figyelemre méltó algoritmusok születtek differenciálegyenletek szimbolikus megoldásainak vizsgálatában is. A Risch-algoritmushoz hasonló döntési eljárás létezik racionálisfüggvény-együtthatójú homogén másodrendű közönséges differenciálegyenlet zárt alakú megoldásainak meghatározására (Kovacic-algoritmus, [41]). Magasabb rendű lineáris esetben az Abramov-eljárás ([1]) a polinomegyüttható egyenletek zárt racionális megoldásait, a Bronstein-algoritmus ([5]) pedig az $\exp(\int f(x)dx)$ alakú megoldásokat határozza meg. Az egyik legújabb eredményt M. van Hoeij adta ([33]), aki minden eddiginél jobb algoritmust talált formális hatványsorok és racionális

függvények feletti lineáris differenciáloperátorok faktorizálására. Parciális differenciálegyenletek esetében a Lie-féle szimmetria-módszerek állnak rendelkezésre.

A faktorizáláson alapuló eljárások komoly jelentőséggel bírnak a komputeralgebrai algoritmusok kutatásában. Olyannyira igaz ez a megállapítás, hogy sokan az egész tudományterület születését Berlekamp azon publikációjától számítják, amelyben hatékony algoritmust ad kis p karakterisztikájú véges testek feletti egyváltozós polinomok faktorizációjára. Ekkor, 1969-ben vált először világossá, hogy a polinomfaktorizáció algoritmikus időbonyolultsága kisebb lehet az egészek faktorizációjánál: két elemű test esetén egy n-edfokú polinom faktorizációja (ahol a polinom egy n bites egésszel reprezentálható) csak $\mathcal{O}(n^3)$, míg az ugyanekkora egészé exponenciális komplexitású. Ez utóbbi tény később a prímség gyors eldönthetőségével párosítva egy másik tudományterület, a modern kriptográfia alapjait vetette meg. Berlekamp eredményét később nagyobb p karakterisztikákra is kiterjesztette. Hogy hasonlóan jó futási időt kapjon, az algoritmusába véletlen elemeket illesztett. Ez volt talán az első olyan algoritmus, ami randomizált elemek felhasználásával polinomiálissá tett egy determinisztikusan exponenciális időbonyolultságú problémát. A mai komputeralgebra rendszerek nagyobb véges testekre is rutinszerűen alkalmazzák Berlekamp eljárását talán anélkül, hogy a felhasználók többségének az algoritmus valószínűségi eredetéről tudomása lenne.

Nemsokkal azután, hogy a véges testek feletti polinomfaktorizáció megoldódott, Zassenhaus van der Waerden 1936-os *Moderne Algebra* könyvét alapulvéve a p-adikus számok aritmetikájának ún. *Hensel-lemmá*ját alkalmazta a faktorok kiterjesztésére. A "Hensel lifting" — ahogy ma az eljárását nevezik — egy általános megközelítési módszer arra, hogyan rekonstruáljuk a faktorokat moduláris képeikből. Az interpolációval ellentétben, ami több képpont meglétét követeli meg, a Hensel lifting csak egyetlen képpontot igényel. Az egész együtthatós polinomok faktorizációjára adott Berlekamp–Zassenhaus-féle algoritmus alapvető jelentőségű, mégis rejteget magában két csapdát. Az egyik, hogy bizonyos fajta polinomokra az algoritmus időkomplexitása exponenciális. Sajnos az algebrai számtestekre épített polinomok faktorizációjánál (a Kronecker-módszer, ami alapvetően az egész polinomok faktorizációjára vezethető vissza) sok ilyen 'rossz' polinom kerül elő. A második, hogy többváltozós polinomokra hasonló reprezentációs probléma lép fel, mint amilyet ritka mátrixok Gauss-eliminációjánál tapasztalunk. Az első problémát egy, a számok geometriáján alapuló diofantoszi optimalizálás, a Lenstra-Lenstra-Lovász nevével fémjelzett ún. rácspont-redukciós algoritmus oldotta meg, amit a Berlekamp-módszerrel együtt használnak. Ezen polinomiális algoritmus mellé még egy olyan eljárás társul, amely arról gondoskodik, hogy a Hensel lifting 'jó' moduláris képről induljon és mikor érjen véget. A többváltozós polinomfaktorizáció említett reprezentációs problémájára is születtek megoldások, közülük az ún. black-box megközelítés tűnik a legjobbnak. Ez a második olyan terület, ahol a randomizálás kritikus szerepet kap a hatékony algoritmusok tervezésében.

Nem szabad azonban túlbecsülni az algoritmusok elméleti komplexitása vizsgálata során kapott eredményeket: a gyakorlatban a Berlekamp–Zassenhaus–Hensel-féle algoritmus hatékonyabban működik, mint az elméletileg jobb Lenstra–

Lenstra–Lovász-féle eljárás. Hasonló jelenség figyelhető meg a nagy egész számok szorzásánál. A Karacuba-algoritmus komplexitása $\mathcal{O}(n^{\log_2 3})$, míg a Schönhage–Strassen-algoritmusé $\mathcal{O}(n \log n \log\log n)$, a komputeralgebrai rendszerek mégis a Karacuba-módszert használják nagypontosságú egészek szorzására, mert a másik módszer előnye csak asztronómiai méretű számoknál mutatkozik meg.

Mivel a komputeralgebra-rendszerek szimbolikusan, teljes pontossággal számolnak, az algoritmusok időbonyulultságának vizsgálatán kívül mindig szükség van a tárbonyolultságuk vizsgálatára is. Tekintsünk egyszerű példaként egy n ismeretlenes n egyenletből álló lineáris egyenletrendszert, ahol minden egyes egész együttható elfér a számítógép ω hosszúságú rekeszében. Könnyű észrevenni, hogy a szokásos egész Gauss-eliminációt alkalmazva a redukció eredményeként kapott együtthatók egyenként $2^{n-1}\omega$ tárhelyet igényelnek. Ha az együtthatók polinomok lennének és polinomaritmetikát használnánk, az eredménypolinomok együtthatóinak mérete, csakúgy mint a fokszámuk exponenciális növekedést mutatna. A megfigyelt exponenciális növekedés ellenére a kapott végeredmény mégis 'normális' méretű, hiszen a Cramer-szabály miatt a megoldások determinánsok hányadosaiként is megkaphatók, amelyek pedig közelítőleg csak $n\omega$ tárhelyet igényelnek. Ezt a jelenséget nevezi az irodalom *intermediate expression swell*nek. Előfordulása gyakori a komputeralgebrai algoritmusokban. Fontos példa a polinomok legnagyobb közös osztójának meghatározására szolgáló algoritmus, amelyet a polinomfaktorizálástól a szimbolikus integrálásig sok más eljárás is használ. A hagyományos módszerek mellett (polynomial remainder sequences) a heurisztikának ez esetben is komoly jelentősége van.

Valójában a heurisztikus, probabilisztikus módszereknek egy új elmélete van születőben a komputeralgebrában egyrészt az intermediate expression swell elkerülésére, másrészt a determinisztikusan magas komplexitású algoritmusok hatékonyságának növelésére. A probabilisztikus algoritmusok esetében a nem megfelelő működésnek is van pozitív valószínűsége, ami vagy az esetleges rossz válaszban nyilvánul meg (Monte Carlo csoport) vagy, habár sohasem kapunk rossz választ, elképzelhető, hogy polinomiális időben nem kapunk semmit (Las Vegas csoport). A már említetteken kívül szép eredményeket értek el heurisztikákkal többek között polinomazonosság tesztelésénél, polinomok irreducibilitásának vizsgálatánál, mátrixok normálformáinak (Frobenius, Hilbert, Smith) meghatározásánál. Szerepük minden valószinűség szerint a jövőben is növekedni fog.

A fejezet eddigi részében a lényegesebb szimbolikus algoritmusok közül csemegéztünk. Korábban említettük, hogy a komputeralgebra rendszerek képesek numerikus számítások elvégzésére is: a hagyományostól eltérően a számítási pontosságot a felhasználó határozhatja meg. Gyakran előfordul, hogy a szimbolikus és numerikus számításokat együtt célszerű használni. Tekintsük például egy differenciálegyenlet szimbolikusan kiszámolt hatványsor megoldását. A csonkított hatványsort ezután bizonyos pontokban a szokásos lebegőpontos aritmetikával kiértékelve a differenciálegyenlet megoldásának numerikus approximációját kapjuk. Amennyiben a megoldandó probléma valamilyen valós fizikai probléma approximációja, a szimbolikus számítások vonzó lehetőségei gyakran érvényüket vesztik; egyszerűen mert túl bonyolultak vagy túl lassúak ahhoz, hogy hasznosak vagy szükségesek legyenek, hiszen a megoldást is nume-

rikusan keressük. Más esetekben, ha a probléma szimbolikusan kezelhetetlen, az egyetlen lehetséges út a numerikus approximációs módszerek használata. Ilyen akkor fordulhat elő, ha a létező szimbolikus algoritmus nem talál zárt megoldást (pl. nem elemi függvények integrálja stb.), vagy nem létezik a problémát kezelni tudó szimbolikus algoritmus. Ámbár egyre több numerikus algoritmusnak létezik szimbolikus megfelelője, a numerikus eljárások nagyon fontosak a komputeralgebrában. Gondoljunk csak a differenciál- és integrálszámítás területére: bizonyos esetekben a hagyományos algoritmusok — integráltranszformációk, hatványsor-approximációk, perturbációs módszerek — lehetnek a legcélravezetőbbek.

A komputeralgebra algoritmusok tervezésénél a jövőben egyre nagyobb szerepet kapnak a párhuzamos architektúrájú számítógépek. Habár sok meglévő algoritmus "ránézésre" párhuzamosítható, nem biztos, hogy a jó szekvenciális algoritmusok párhuzamos számítógépeken is a legjobbak lesznek: az optimum talán egy teljesen különböző eljárással érhető el.

4. A komputeralgebra-rendszerek jelentősége, alkalmazásaik

A komputeralgebra-rendszerek használatának legalább három oka lehet: olyan szimbolikus számításokat szeretnénk végezni, amelyek

- papírceruza-módszerrel is végrehajthatók, de számítógéppel gyorsabban és "félreszámolás"-mentesen,
- papírceruza-módszerrel reménytelennek tűnnek, de számítógéppel a meglévő algoritmusoknak és matematikai technikáknak köszönhetően többékevésbé rutinszerűen hajthatók végre,
- komoly előkészületeket és erőfeszítéseket igényelnek még számítógép használatával is. Az ilyen jellegű problémák számítógép nélkül általában megoldhatatlanok. Fennmarad tehát a jobb és gyorsabb algoritmusok keresésének feladata annak minden nehézségével és szépségével együtt.

A szimbolikus számítások igénye nagyon sok tudományterület problémáinak megoldása során felmerül, alkalmazások széles rétege előtt nyitva ezzel új fejezetet. A komputeralgebrát ez a "határtudomány" pozíciója kutatói szempontból mélységgel és gazdagsággal, de ugyanúgy nehézséggel és változékonysággal is felruházza. Az eddig elmondottakból kitűnik, hogy ezek a rendszerek nemcsak tudományos kutatók és elméleti szakemberek, hanem mérnökök és ipari kutatócsoportok munkatársai számára is jól használhatóak.

"Mathematics is a basis of technological progress, and technological progress is a key for international competitiveness. Automating an important part of the mathematical program-solving process is a key technology for a nation that wishes to control, structure, and accelerate technological progress. The automation of the solution of mathematical problems is a powerful lever by which human

productivity and expertise can be amplified many times." [4]

A világ vezető államai felismerve a komputeralgebra-rendszerek nemzetgazdasági jelentőségét egyre növekvő mértékű támogatást nyújtanak oktatásához, tudományos és ipari kutatóműhelyek létrehozásához, a tudományterület népszerűsítéséhez. A világ számos egyetemén oktatják a matematika alapjait valamilyen általános célú komputeralgebrai program (általában Maple és Mathematica) segítségével, így a matematika absztrakt objektumai 'kézzelfoghatóvá' válnak, az új gondolatokat, tételeket és azok alkalmazásait azonnal ki is lehet próbálni. Az elmúlt években hazánk számos egyetemén és főiskoláján indult meg a komputeralgebra rendszerek oktatási keretek közé illesztése. Az Eötvös Loránd Tudományegyetem Komputer Algebra Tanszékének gondozásában az alkalmazásokon kívül megismerhető ezen rendszerek matematikai és informatikai háttere is.

Az alábbiakban megemlítünk néhány olyan területet, ahol komputeralgebra-rendszerek alkalmazása révén komoly eredmények születtek:

- matematika — algebra, algebrai geometria, analízis, bifurkációelmélet, csoportelmélet, geometria, kriptográfia, kódelmélet, kombinatorika, numerikus analízis, számelmélet, topológia, valószínűségelmélet,
- fizika — asztrofizika, atomfizika, relativitáselmélet, égi mechanika, folyadék mechanika, kvantumelektro- és kromodinamika, kvantummechanikai perturbációelmélet, nagyenergiájú fizika, nukleáris mágneses rezonanciavizsgálat, optika, plazmafizika, szilárdtestfizika,
- kémia — biokémia, gyógyszeripar, reakcióegyenletek analízise, molekulák struktúraanalízise, komplex reagáló rendszerek stabilis egyensúlyának vizsgálata, molekulák térbeli mozgásának mechanikája,
- mérnöki tudományok — antennatervezés, elektronikus hálózatok analízise, geostatisztika, hajótesttervezés, helikopterrotor-tervezés, hidrodinamika, kép- és jelfeldolgozás, kontrollelmélet, radartervezés, robotika, turbinatervezés stb.

A szimbolikus matematika egyik legérdekesebb felhasználói területe a matematika önmaga.

"I see the main role of symbolic algebra systems as that of helping to formulate hypotheses, search for examples and counterexamples, and in general explore ramifications of mathematical models. In other words, the main role of these systems is to obtain mathematical insight. Once that insight is obtained,

[4] A.C. Hearn, Ann Boyle, B.F. Caviness. *Future Directions for Research in Symbolic Computation*, Siam Reports on Issues in the Mathematical Sciences, Philadelphia, 1990.

one can then go on and construct canonical mathematical proofs, in which there might not even be any traces of the use of computer algebra." [5]

Két, a tanszékünk témáiból vett példával szeretném illusztrálni a szimbolikus számítások hasznosságát az elméleti matematikában.

Komplex halmazok

Bizonyos fajta szimmetria-tulajdonságokkal rendelkező, kvadratikus iterációból származó komplex halmazok vizsgálatára irányul az első probléma. Adjuk meg az összes olyan Γ halmazt, amire

$$\Gamma \subset \mathbb{C} \setminus \mathbb{R},$$

$$\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_N\} = \{-\gamma_1, -\gamma_2, \ldots, -\gamma_N\} = -\Gamma,$$

$$\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_N\} = \{\overline{\gamma}_1, \overline{\gamma}_2, \ldots, \overline{\gamma}_N\} = \overline{\Gamma},$$

$$\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_N\} = \{R(\gamma_1), R(\gamma_2), \ldots, R(\gamma_N)\} = R(\Gamma),$$

ahol $R(x) = Ax^2 + Ax + E$, $A, E \in \mathbb{R}$, $A, E \neq 0$, $\gamma_i \neq \gamma_j$ $\forall i \neq j$ és amelynek egyetlen valódi részhalmaza sem elégíti ki az iménti feltételeket.

Sejtés: $\Gamma = \{\sqrt{E}, -\sqrt{E}\}, E < 0$.

A szóbajöhető Γ-kat vizsgálva, $\mathrm{card}(\Gamma) \leq 100$-ra a sejtés a szimmetrikus polinomok Newton–Girard-formuláit és a rezultáns módszert használva számítógéppel viszonylag gyorsan verifikálható, míg "kézzel" számolva a $\mathrm{card}(\Gamma) = 8$ eset is komoly munkát igényel. Az általános bizonyítás nehéznek tűnik [42]. A feladat természetéből adódóan numerikus számítási módszerek használata itt nem realizálható.

Teljesen additív függvények és kongruenciák

Jelöljön $P(z) := 1 + A_1 z + A_2 z^2 + \ldots + A_k z^k (k \geq 1)$ egy valós együtthatós polinomot, $P(z) \notin \mathbb{Q}[z]$. Jelölje továbbá E a végtelen sorozatok lineáris terének $Ez_n := z_{n+1}$ operátorát. A $P(E) = I + A_1 E + \ldots + A_k E^k$ operátort az $f(n)$ függvényre alkalmazva kapjuk, hogy $P(E)f(n) = f(n) + A_1 f(n+1) + \ldots + A_k f(n+k)$.

Sejtés: Legyen $k \in \mathbb{N}$ rögzített. Ha egy teljesen additív valós értékű f függvény $\forall n \in \mathbb{N}$ esetén kielégíti a $P(E)f(n) \equiv 0 \pmod 1$ kongruenciaegyenletet, akkor $f(n) = 0$ $\forall n \in \mathbb{N}$ esetére.

Kis k értékekre $(k = 2, 3, 4)$ a bizonyítás két részből áll. Először be kell látni, hogy az állításban foglaltak igazak valamilyen $n \leq N$ korlátra, aztán ez alapján a második lépésben $\forall n \in \mathbb{N}$-re. A sejtés bizonyításának első része már $k = 4{,}5{,}\ldots$

[5] A.M. Odlyzko, *Applications of symbolic mathematics to mathematics*, Applications of Computer Algebra, Kluwer Academic Press, Boston, 1985.

esetében is elképzelhetetlen lenne szimbolikus program használata nélkül. Megjegyzendő, hogy $k = 5, 6, \ldots$ -ra teljes bizonyítás nem ismeretes, az általános bizonyítás ($\forall k$-ra) itt is nehéznek tűnik [43].

A következő példával a szimbolikus programozási nyelvek gyakorlati jelentőségét szeretném bemutatni. A modellben a futóműnek a vázhoz való felfüggesztésén keresztül a kényelmes, zötyögésmentes autózás lehetőségeit vizsgáljuk[6].

Lengéscsillapítók az autóiparban

Jelölje m az autó tömegét, k a felfüggesztésnél használt rugóra jellemző állandót és b a lengéscsillapító minőségére jellemző, a lengéscsillapítás erősségét megadó konstans értéket, valamint $x(t)$ a t-edik időpillanatban az autó karosszériájának az úthoz viszonyított magasságát. Ekkor a csillapított rezgés jelenségének egyenletét felírva az alábbi állandó együtthatós, homogén másodrendű lineáris differenciálegyenletet használhatjuk: $mx''(t) + bx'(t) + kx(t) = 0$.

a) Adjuk meg az egyenlet általános megoldását attól a pillanattól kezdve, amikor az autó egy váratlan gödörbe kerülve v_0 m/s függőleges irányú sebességre tesz szert, miközben a karosszéria y méterrel kerül közelebb az úthoz.

b) Rajzoljuk ki a karosszéria úthoz viszonyított magasságának a változását az első 15 másodpercben, ha $m = 1000\ kg, b = 300\ kg/s, k = 2500\ kg/s^2, v_0 = 20\ cm/s, y = 8\ cm$.

c) Mekkora lesz a kilengés 5 másodperc múlva? Mekkora a maximális kilengés?

d) Hogyan függ a kilengés az autó tömegétől? Vizsgáljuk meg a kérdést az előbbi adatokkal, az autó tömege legyen 500 és 1500 kg közötti.

e) Mekkora lenne a minimális értéke a lengéscsillapítási konstansnak, ha teljesen sima, (elméletileg) "ugrálásmentes" utazást szeretnénk?

Először oldjuk meg a differenciálegyenletet a megadott kezdeti érték feltételekkel:

```
> ODE:= m*diff(x(t),[t$2]) + b*diff(x(t),t)+k*x(t) = 0;
```

$$ODE := m\,(\frac{\partial^2}{\partial t^2}\,\mathrm{x}(t)) + b\,(\frac{\partial}{\partial t}\,\mathrm{x}(t)) + k\,\mathrm{x}(t) = 0$$

```
> kezd_felt:=x(0)= y, D(x)(0) = v[0]:
> mego:=dsolve({ODE,kezd_felt},x(t));
```

$$mego := \mathrm{x}(t) = \frac{1}{2}\,\frac{(b\,y + \sqrt{b^2 - 4\,m\,k}\,y + 2\,v_0\,m)\,e^{(-1/2\,\frac{(b-\sqrt{b^2-4\,m\,k})\,t}{m})}}{\sqrt{b^2 - 4\,m\,k}}$$
$$- \frac{1}{2}\,\frac{(b\,y - \sqrt{b^2 - 4\,m\,k}\,y + 2\,v_0\,m)\,e^{(-1/2\,\frac{(b+\sqrt{b^2-4\,m\,k})\,t}{m})}}{\sqrt{b^2 - 4\,m\,k}}$$

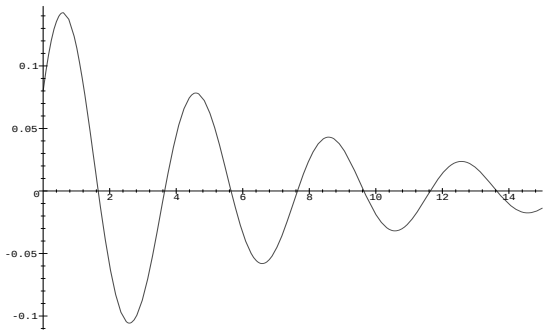
[6] A használt programnyelv a MapleV Release 4.0b

Rajzoljuk ki a kilengést az első 15 másodpercben:

```
> f:=unapply(subs(m=1000,b=300,k=2500,v[0]= 0.2,y=0.08,
> rhs(mego)),t);
```

$$f := t \to -\frac{1}{19820000}\left(424.00 + .08\sqrt{-9910000}\right)\sqrt{-9910000}\, e^{\left(-1/2000\left(300-\sqrt{-9910000}\right)t\right)} + \frac{1}{19820000}\left(424.00 - .08\sqrt{-9910000}\right)\sqrt{-9910000}\, e^{\left(-1/2000\left(300+\sqrt{-9910000}\right)t\right)}$$

```
> plot(f,0..15);
```



A kilengés az 5. másodpercben:

```
> x(5)= evalf(f(5));
```

$$\mathrm{x}(5) = .063007\,18586$$

A zökkenés pillanatától a maximális kilengésig eltelt idő, és abban a pillanatban a kilengés mértéke:

```
> fsolve(diff(f(t),t),t,0..2);
```

$$.5970840377$$

```
> x(")=evalf(f("));
```

$$\mathrm{x}(.5970840377) = .1425887428$$

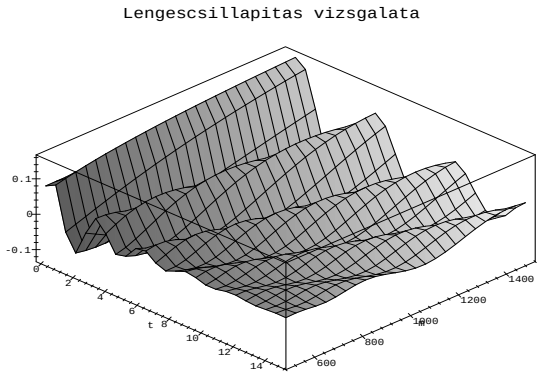
A kilengés vizsgálata az eltelt idő és az autó tömegének függvényében:

```
> g:=unapply(subs(b=300,k=2500,v[0]= 0.2,y=0.08,rhs(mego)),t,m);
```

$$g := (t,\, m) \to \frac{1}{2}\,\frac{\left(24.00 + .08\sqrt{90000 - 10000\,m} + .4\,m\right) e^{\left(-1/2\,\frac{\left(300-\sqrt{90000-10000\,m}\right)t}{m}\right)}}{\sqrt{90000 - 10000\,m}}$$

$$-\frac{1}{2}\,\frac{(24.00-.08\,\sqrt{90000-10000\,m}+.4\,m)\,e^{(-1/2\,\frac{(300+\sqrt{90000-10000\,m})\,t}{m})}}{\sqrt{90000-10000\,m}}$$

```
> plot3d(g(t,m),t=0..15,m=500..1500,axes=BOXED,orientation=
> [-45,35],style=PATCH,title='Lengéscsillapítás vizsgálata');
```



Ismert, hogy a rezgés pontosan akkor osszcillációmentes, ha a differenciálegyenlet karakterisztikus egyenlete gyökei valósak. A minimális lengéscsillapítási konstanst akkor kapjuk, ha az említett egyenlet diszkriminánsa nulla, vagyis ha

```
> b=sqrt(4*1000*2500);
```

$$\mathrm{b} = 1000\,\sqrt{10}$$

Figyeljük meg, hogy az 'ugrálásmentes' utazáshoz az eredetinél több, mint tízszer erősebb lengéscsillapító kell:

```
> evalf(rhs(''))/300;
```

$$10.54092553$$

Számításaink során bizonyos esetekben szimbolikus, máskor numerikus módszereket használtunk, mindig a legkézenfekvőbbet alkalmazva.

5. Korlátok és a jövő perspektívái

A jövőbe tekintő spekulációk általában a közelmúlt relatíve biztonságos extrapolációi, esetleg valamilyen kevésbé megbízható trendek és lehetőségek kockázatos előrevetítései. Egy fiatal, gyorsan változó tudományterület, mint például a komputeralgebra esetében bármiféle előrejelzés nehéz, hiszen a számítástudomány önmagában is viharos sebességgel fejlődő ágazat. A kereskedelmi szektor agresszívan keresi a lehetőséget a szimbolikus szoftverek terjesztésére, egyre több kutatóhoz, mérnökhöz, diákhoz juttatva el így őket. Ezt egybevetve az előző

fejezetben mutatott alkalmazhatósági lehetőségekkel, fogalmat alkothatunk a fejlődés várható dinamikájáról. Az alábbiakban az eddigi hiányosságokat, korlátokat valamint a lehetséges igényeket szeretném összefoglalni, talán a jövő komputeralgebra-rendszereit vetítve ezzel előre.

1. Szimbolikus vagy numerikus számítások? Annak a ténynek, hogy a komputeralgebra-rendszerek tetszőleges pontosságú számítások elvégzésére képesek, van egy negatív oldala is: a használt saját (szoftveres) lebegőpontos aritmetika miatt a gépi pontosságot meghaladó numerikus műveletek százszor, ezerszer lassabbak lehetnek, mint más, beépített (hardveres) aritmetikát használó programok esetében. Ezért numerikus problémáknál mindig felvetődik a kérdés: valóban szükség van-e racionális számokkal történő egzakt vagy tetszőleges pontosságú lebegőpontos aritmetikára? Ellenkező esetben célszerűbb lehet valamely tradicionális programozási nyelv használata.

2. Szintaxis és szemantika. Ami a szintaxist és a szemantikát illeti, a szimbolikus nyelvek programozása nehezebb, mint a hagyományos numerikus nyelveké. Tengernyi beépített függvényt tartalmaznak, amelyeknek óvatlan használata váratlan eredményeket produkálhat. Sok esetben ezért nem árt, ha ismerjük a rendszerek és eljárásaik belső működését, az objektumok ábrázolásmódját; így jobban kordában tarthatjuk méretüket, elkerülhetjük a nemkívánt intermediate expression swell jelenséget. A szimbolikus nyelvek használata során elkövetett programozási hibák nyomkövetése szintén újszerű, nem minden esetben könnyű feladat.

3. Memóriaigény és futási idő. Általában nehéz megbecsülni az adott probléma megoldásához szükséges számítások idő- és tárkomplexitását. A legmegfelelőbb matematikai modell kiválasztása és hatékony programozása, a számítások optimalizálása nagy türelmet és szakértelmet kíván (de mindig megéri a fáradságot). Mindezeken kívül a szimbolikus számítások a numerikustól eltérő gondolkodásmódot igényelnek. Bizonyos esetekben ügyes változóhelyettesítéssel drasztikusan redukálhatjuk valamely algoritmus idő- és tárigényét, vagy éppen megoldást találhatunk egy olyan problémára, ami enélkül megoldhatatlannak tűnik.

4. Kifejezések formátuma. A szimbolikus rendszerek használata során gyakran vetődik fel a kérdés: mi az adott kifejezés legegyszerűbb alakja, hogyan érdemes azt egyszerűsíteni? Az $(x-1)^2(x+1)^3(2x+3)^4$ kifejezés sokkal tömörebbnek és kifejezőbbnek tűnik, mint a $16x^9 - 80x^8 + 88x^7 + 160x^6 - 359x^5 + x^4 + 390x^3 - 162x^2 - 135x + 81$ kifejezés, ez utóbbi viszont jobban használható, ha pl. az x^6 együtthatójára vagyunk kíváncsiak. A kiterjesztett vagy faktorizált kifejezések problémájára két másik illusztris példa:

- $x^{1000} - 1$ és $(x-1)(x^{999} + x^{998} + \ldots + x + 1)$.
- $(x+1)^{1000}$ és $x^{1000} + 1000x^{999} + \ldots + 1000x + 1$.

Az absztrakt objektumok képernyőn való megjelenítése pszichológiai problémákat feszeget: ámbár konverzióra mindig van lehetőség, nem minden esetben látszik előre, mi lesz az ember számára a legegyszerűbb, legjobban érthető forma.

(Pl. egy- és többváltozós polinomok, tenzorok, gráfok stb.) Másrészt egy jól felépített rendszerben az input és output kifejezések formátuma nem térhet el nagyon a standard matematikai jelölésrendszertől.

5. Tervezési kérdések. Az egyik legfontosabb tervezési probléma az egyszerűsítések automatizálásának kérdése. A MAPLE például a $0 * f(1)$ szorzatot automatikusan nullára egyszerűsíti, ami helytelen, ha $f(1)$ nem definiált vagy végtelen. A tervezőknek egyensúlyt kell találniuk a matematikai korrektség és a használhatóság, hatékonyság között.

6. Típusok. Ahhoz, hogy az ábrázolt objektumoknak a pontos jelentését ismerjük, tudnunk kell, hogy milyen értelmezési tartományhoz tartoznak. Ezt nevezzük az objektum típusának. A típus azonban nem mindig egyértelmű. A 29 szám nemcsak egész, de nemnegatív sőt pozitív egész. Általában a felhasználó nem akarja minden lépésnél explicit módon meghatározni a használt objektumok típusát, a rendszernek kell ezt megtennie, amennyire ez lehetséges. A típust nem mindig lehet automatikusan felismerni. Amikor a típus már valamilyen módon meghatározott, a megfelelő műveletek alkalmazhatók. Például a rendszer felismeri, hogy lánctört-együtthatós polinomok szorzása kommutatív, mátrix együtthatós polinomoké viszont nem. Ily módon a megfelelő tartományok hierarchiája építhető fel, ahogy ezt az AXIOM nyelv készítői tették.

7. Automatikus következtetés. Tekintsük az alábbi, AXIOM szintaxisban megadott szabályokat:

```
sinCosProducts == rule
  sin(x) * sin(y) == (cos(x-y) - cos(x+y))/2
  cos(x) * cos(y) == (cos(x-y) + cos(x+y))/2
  sin(x) * cos(y) == (sin(x-y) + sin(x+y))/2
```

A szabályok bal oldalát mintának, a jobb oldalát a helyettesítésének nevezzük. Amikor egy szabályt alkalmazunk, egy ún. mintaillesztő eljárás végigpásztázza az argumentumok részkifejezéseit, és az első illeszkedő mintát helyettesíti a szabály jobb oldalán lévő kifejezéssel. Ez a paradigma nagyon hasznos és fontos például kifejezések különböző formái közötti transzformációk végrehajtása során. Adott szabályrendszer esetén a szabályok egymás utáni alkalmazásakor természetszerűleg vetődik fel a terminálás kérdése. Ebben az általánosságban azonban ez egy eldönthetetlen probléma (a Turing-gépek megállási problémája visszavezethető a terminálás problémájára). Bizonyos esetekben a Knuth–Bendix-féle eljárás ([40]) alkalmazható, de olyan eset is lehetséges, amikor bizonyíthatóan nem konstruálható olyan teljes rendszer, amely az adott szabályhalmazzal ekvivalens lenne. Szorosan idekapcsolódó kérdéskör az automatikus következtetés és tételbizonyítás problémaköre. Az eddigi biztató eredmények ellenére hosszú még az út az automatikus tételbizonyítók mindennapos használatáig. Mindezek ellenére ezen sorok írója meggyőződéssel hiszi, hogy a jövő komputeralgebra-rendszerei képesek lesznek tanulásra, interaktív működésük alatt funkcionalitásuk változtatására, tételbizonyításra, egyszóval mindarra, amit ma mesterséges intelligenciának nevezünk.

8. Megoldhatóság. A komputeralgebra-rendszerekben felhalmozott hatalmas ismeretanyag ellenére tisztában kell lennünk azzal, hogy a matemati-

kának sok olyan területe van még, ahol a mai rendszerek nem sok segítséget tudnak nyújtani vagy a hatékony algoritmusok hiánya vagy egész egyszerűen a megfelelő megoldási módszerek ismeretének hiánya miatt. Ilyen, a kutatások élvonalába tartozó területek pl. a parciális differenciálegyenletek, a nem elemi (pl. Bessel-féle) függvényeket tartalmazó határozatlan és határozott integrálok, felületi integrálok, nemkommutatív algebrák stb. Léteznek olyan problémák is, amelyekre bizonyítható, hogy nem létezik egzakt szimbolikus megoldás. Tekintsük például a transzcendens kifejezések azon TK osztályát, amelyek egy x változóból, racionális konstansokból, a transzcendens konstans π -ből, és a +,*,sin,abs függvényszimbólumokból épülnek fel. Két kifejezés pontosan akkor ekvivalens, ha $\mathbb{R}$-en ugyanazt a függvényt írják le. Richardson és Matijasevic Hilbert 10. problémájának eldönthetetlenségére adott bizonyítását alapulvéve Caviness bizonyította be, hogy a TK osztálybeli kifejezések ekvivalenciaproblémája eldönthetetlen [11].

9. Implementáció. Egy adott algoritmushoz tartozó legjobb implementációt megtalálni nem minden esetben könnyű vállalkozás. Az objektumok adatábrázolásától, a választott programnyelv lehetőségeitől függően több "legjobb" implementáció is lehetséges. Ezenkívül a különféle speciális esetek gyakran különböző megközelítési módot igényelnek. A használhatóságot tekintve az informatika legújabb eredményeinek alkalmazása megkönnyítheti az egyensúlyozást a hatékony kód és a kényelmes használat között.

10. Párhuzamosítás. A parallelizálás irányába történt erőfeszítések várhatólag előbb-utóbb meghozzák gyümölcsüket. Az eddigi eredmények ígéretesek.

11. Modulok és könyvtárak. Ma már a komputeralgebra-rendszerek lényegesen nyitottabbak, vagyis az egyik rendszerből meghívható több másik szimbolikus vagy numerikus nyelven megírt programrészlet. Ezeket a programegységeket (modulokat) a felhasználói felületen keresztül interaktívan csatolhatjuk. A modulkönyvtárak közös használatához a be- és kimenetet szabványosítani kell. Ennek egy megoldásán dolgozik az OpenMath tervezet [55].

12. Felhasználói felület. A legtöbb általános célú komputeralgebra-rendszer rendelkezik valamilyen grafikus felülettel. Ami pedig fejlődésüket illeti, véleményem szerint folytatódni fog az a napjainkban is érzékelhető pozitív tendencia, ami elvezet a szabványos elemekből építkező (ábra, szöveg, formula, stb), dokumentumkészítésre alkalmas, internet- és hypertext-lehetőséggel felvértezett univerzális programcsomagokhoz.

A fejezetben látott hiányosságok ellenére a komputeralgebra-rendszerek nagyon gyakran szolgálnak kellemes meglepetésekkel, tudásuknál fogva, valamint interdiszciplináris természetükből adódóan reményteljes és sokatígérő jövő előtt állnak. Megjelenésükkel új módszerek, új eszközök léptek színre, amelyek fokozatosan megváltoztatják gondolkodásmódunkat, modellalkotó képességünket. Segítségükkel a problémák új megvilágításba kerülnek, a természettudományi és matematikai törvények egyre apróbb és apróbb részletei kristályosodnak ki, ezáltal újabb és újabb felfedezéseket tehetünk a csodák — a matematikai struktúrák univerzumában.

6. Irodalmi összefoglaló

A komputeralgebráról és a kapcsolódó tudományterületekről számos publikáció és könyv született. Az érdeklődőknek sorolunk fel az összefoglaló matematikai jellegű munkák közülük néhányat: Caviness [10], Davenport et al. [18], Geddes et al. [22], Knuth [39], Mignotte [49], Mishra [51], Pavelle et al. [56], Winkler [64].

További információk lelhetők a komputeralgebra informatikai oldaláról az alábbiakban: Buchberger [6], Christensen [15], Gonnet és Gruntz [24], Harper et al. [29], Miola [50] valamint a világhálózaton, pl. [66].

Az alkalmazásokról könyvek és cikkek nagyon széles választéka áll rendelkezésre, pl. Akritas [2], Cohen et al. (ed.) [16, 17], Grossman (ed.) [28], Hearn (ed.) [31] és Odlyzko [53].

A komputeralgebra-rendszerek oktatásban betöltött szerepéről lásd pl. Karian [37] és Uhl [62] munkáját.

Ajánlott konferenciakiadványok: AAECC, DISCO, EUROCAL, EUROSAM, ISSAC és SYMSAC.

Komputeralgebrai folyóiratok: *Journal of Symbolic Computation (JSC)* - Academic Press, *Applicable Algebra in Engineering, Communication and Computing (AAECC)* - Springer-Verlag, *SIGSAM Bulletin* - ACM Press.

Irodalom

[1] S.A. Abromov, K. Kvashenko. *Fast Algorithms to Search for the Rational Solutions of Linear Differential Equations with Polynomial Coefficients*, Proceedings of ISSAC '91, ACM Press, 1990, 267–270.

[2] A.G. Akritas. *Elements of computer algebra with applications*, Wiley, New York, 1989.

[3] D. Barton and J.P. Fitch. Camal: *The Cambridge Algebra System*, SIGSAM Bull., **8**(3), 1974, 17–23.

[4] C. Batut, D. Bernardi, H. Cohen and M. Oliver. *User's Guide to* Pari-GP, *version 1.39.* Université Bordeaux I, 1995.

[5] M. Bronstein. *Linear Ordinary Differential Equations: breaking through the order 2 barrier*, Proceeding of ISSAC '92, ACM Press, 1992, 42–48.

[6] B. Buchberger, G.E. Collins, M. Encarnación, H. Hong, J. Johnson, W. Krandick, R. Loos, A. Mandache, A. Neubacher, H. Vielhaber. *SACLIB 1.1 user's guide* Techn. Rep. RISC Linz, 1993, 93–19.

[7] G. Butler, J. Cannon. *The Design of Cayley - A Language for Modern Algebra*, Design and Implementation of Symbolic Computation Systems, Lecture Notes in Computer Science **429**, Springer, 1990, 10–19.

[8] J. Cannon, W. Bosma. CAYLEY *Quick Reference Guide*, Univ of Sydney, 1991.

[9] A. Capani, G. Niesi. CoCoA *User's Manual*, Univ. of Genova, CoCoA version 3.0b, 1995.

[10] B.F. Caviness. *Computer algebra: past and future*, J. Symb. Comput., **2**, 1986, 217–263.

[11] B.F. Caviness, R.J. Fateman. *Simplification of radical expressions*, Proc. 1976 ACM Symp. on Symbolic and Algebraic Comp. (ed. R.D. Jenks), ACM Press, New York, 1976, 329–338.

[12] B.W. Char, K.O. Geddes et al. *Maple V Library Reference Manual.* Springer, 1991.

[13] B.W. Char, K.O. Geddes et al. *Maple V Language Reference Manual.* Springer, 1991.

[14] B.W. Char, K.O. Geddes et al. *First Leaves: A Tutorial Introduction.* Springer, 1992.

[15] S.M. Christensen. *Resources for Computer Algebra*, Computers in Physics, **8**, 1994, 308–315.

[16] A.M. Cohen et al. (ed.) *Computer Algebra for Industry: Problem Solving in Practice*, Wiley & Sons, 1993.

[17] A.M. Cohen et al. (ed.) *Computer Algebra for Industry 2: Problem Solving in Practice*, Wiley & Sons, 1995.

[18] J.H. Davenport, Y. Siret, E. Tournier. *Computer Algebra: systems and algorithms for algebraic computation*, Academic Press, 1988.

[19] I. Frick. SHEEP *User's Manual*, Univ. of Stockholm, 1977.

[20] Fuchsteiner et al. *MuPAD: Multi Pocessing Algebra Data Tool; Benutzerhandbuch, MuPAD Version 1.1.*, Birkhäuser, 1993.

[21] Fuchsteiner et al. *MuPAD: Multi Pocessing Algebra Data Tool; Tutorial*, Birkhäuser, 1993.

[22] K.O. Geddes, S.R. Czapor, G. Labahn. *Algorithms for Computer Algebra*, Kluwer, 1992.

[23] A. Giovini, G. Niesi. *CoCoA: A User-Friendly System for Commutative Algebra*, Design and Implementation of Symbolic Computation Systems, Lecture Notes in Computer Science **429**, Springer, 1990, 20–29.

[24] G.H. Gonnet, D.W. Gruntz. *Algebraic manipulation: systems*, Encyclopedia of computer science, 3rd edn., Van Nostrand Reinhold, New York, 1993.

[25] J. Graf v. Schmettow. KANT - *a Tool for Computations in Algebraic Number Fields*, Computational Number Theory, 1991, 321–330.

[26] J.W. Gray. *Mastering Mathematica (Programming Methods and Applications)*, Academic Press, Boston, 1994.

[27] D.R. Grayson. *Macaulay 2.*, http://www.math.uiuc.edu/Macaulay2, 1995.

[28] R. Grossman (ed.). *Symbolic Computation: Applications to Scientific Computing*, Frontiers in Applied Mathematics, 5, Soc. for Industrial and Applied Mathematics, Philadelphia, 1989.

[29] D. Harper, C. Wooff, D. Hodginson. *A guide to computer algebra systems*, Wiley, Chichester, 1991.

[30] A.C. Hearn. REDUCE *User's Manual.* The Rand Coorporation, Santa Monica, California, 1987.

[31] A.C. Hearn, A. Boyle, B.F. Caviness. *Future Directions for Research in Symbolic Computation*, SIAM Reports on Issues in the Mathematical Sciences, Philadelphia, 1990.

[32] A. Heck. *Introduction to Maple.* 2nd ed., Springer, 1996.

[33] Mark von Hoeij. *Factorization of Linear Differencial Operators*, Ph.D. thesis, Univ. of Nijmegen, 1996.

[34] C. Hollinger, P. Serf. SIMATH — *a Computer Algebra System*, Computation Number Theory, 1991, 331–342.

[35] L. Hornfeldt. STENSOR *Reference Manual*, Univ. of Stockholm, 1988.

[36] R.D. Jenks, R.S. Sutor. AXIOM, *The Scientific Computation System*, Springer, 1992.

[37] Z.A. Karian (ed.). *Symbolic computation in undergraduate mathematics education*, Math. Assoc. Amer. Notes, **24**, 1992.

[38] Klincsik M., Maróti Gy. *Maple 8 tételben*, Novodat, Szeged, 1995.

[39] D.E. Knuth. *A számítógépprogramozás művészete, 2. kötet, Szeminumerikus algoritmusok*, Műszaki Kiadó, Budapest, 1987.

[40] D.E. Knuth, P. Bendix. *Simple Word Problems in Universal Algebras*, Computational Problems in Abstract Algebra (ed. by J. Leech), Pergamon Press, 1970, 263–279.

[41] J. Kovacic. *An Algorithm for Solving Second Order Homogeneous Differential Equations*, J. Symbolic Computation, **2**(1), 1986, 3–43.

[42] Kovács A. *Sets of Complex Numbers generated by a Polynomial Functional Equation*, Annales Univ. Sci. Bud. Sect. Comp., megj. alatt.

[43] Kovács A., B.M. Phong. *On completely additive functions satisfying a congruence*, megj. alatt.

[44] M.A. van Leeuwen, A.M. Cohen, B. Lisser. *LiE: A Package for Lie Group Computations.* CAN Expertise Center, 1992.

[45] M. MacCallum, J. Skea. SHEEP, *a computer algebra system for general relativity*, Algebraic Computing in General Relativity, Clarendon, 1994, 1–172.

[46] M. MacCallum, F. Wright. *Algebraic computing with Reduce*, Clarendon, Oxford, 1991.

[47] MACSYMA *User's Guide, System Reference Manual, Mathematics Reference Manual*, Macsyma Inc., 1992.

[48] R. Maeder. *Programming in Mathematica*, 2nd. edn., Addison-Wesley, Redwood City, CA, 1991.

[49] M. Mignotte. *Mathematics for Computer Algebra*, Springer, 1992.

[50] A. Miola (ed.). *Design and implementation of symbolic computation systems*, Lecture Notes in Computer Science, **722**, Springer, New York, 1993.

[51] B. Mishra. *Algorithmic Algebra*, Springer, 1993.

[52] Molnárka Gy., Gergó L., Wettl F., Horváth A., Kallós G. *A MapleV és alkalmazásai*, Springer Hungarica Kft., Budapest, 1996.

[53] A.M. Odlyzko. *Applications of symbolic mathematics to mathematics*, Applications of Computer Algebra, Kluwer Academic Press, Boston, 1985.

[54] G.J. Oldenborgh. *An Introduction to Form*, Univ. of Leiden, http://www.can.nl/SystemsOverview/General/Form/.

[55] http://www.can.nl/~abbott/OpenMath.

[56] R. Pavelle, M. Rothstein, J. Fitch. *Computer Algebra*, Scientific American, **245**, 1981, 102–113.

[57] M. Schoenert et al. *GAP: Groups, Algorithms and Programming*, RWTH Aachen, 1994. GAP Manual Release 3.4, http://www.math.rwth-aachen.de:/GAP/Manual/.

[58] Szili L., Tóth J. *Matematika és Mathematica*, ELTE Eötvös Kiadó, Budapest, 1996.

[59] M. Stillman, D. Bayer. *Macaulay User Manual*, 1989.

[60] D. Stoutemeyer. *Derive User Manual*, Soft Warehouse Inc., Honolulu, Hawaii, 1994.

[61] H. Strubbe. *Manual for* SCHOONSHIP, Comput. Phys. Commun., **8**, 1974, 1–30.

[62] J.J. Uhl. MATHEMATICA *and me*, Notices AMS, **35**, 1988.

[63] J.A.M. Vermaseren. *Symbolic Manipulation with* FORM, CAN Expertise Center, 1991.

[64] F. Winkler. *Polynomial Algorithms in Computer Algebra*, Springer, 1996.

[65] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, 1991.

[66] Computer Algebra Web Servers: www.can.nl, www.uni-karlsruhe.de, info.risc.uni-linz.ac.at, medicis.polytechnique.fr, symbolicnet.mcs.kent.edu.

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMELMÉLETI KUTATÓCSOPORT
EÖTVÖS LORÁND TUDOMÁNYEGYETEM
KOMPUTER ALGEBRA TANSZÉK
1088 BUDAPEST, MÚZEUM KRT. 6-8.
e-mail: attila@compalg.elte.hu