A *nagypontosságú aritmetikák* alkalmazása *nevezetes (irracionális) számok* kiszámítására

Szlávi Péter

1991.

0. Témabevezető példa

[α/254: Sharp (1651-1742) könyvelő és táblázatszámoló; π 72 jegy]
[α/115-131: Mersenne, Leibniz, Gregory, ...]

1. A tartalomról, célokról

Nem matematika, hanem főleg algoritmusok és problémák...

2. A $\sqrt{2}$

[δ/72-75,ε/98-99]

A gyök-kettő –érdekes módon– nem tartozott a korai matematika kedvenc, vizsgált konstansai közé. Nagyon hamar napirendre tértek közelítő értékei felett. (Talán ennek oka, hogy már a sumérok is ismerték a négyzetgyökvonás iterációs módszerének formuláját: $\sqrt{2} \approx (a+2/a)/2$ [a=2-re alkalmazva]; bár meglehetősen naiv alapon indokolva.) A nem racionális voltát is a pithagoreusi időben bizonyították. (Eukleidész *Elemek* c. művének X. könyvében bizonyítja, hogy a négyzet átló- és oldalhossza összemérhetetlen.)

2.1. Iterációs módszer ($x_{k+1}=(x_k/2+1/x_k)$)

[β/205,γ/129-132]

Az iterációs módszerekről:

1. Ha az $f(x)$ függvényhez az f'' létezik, és sem f', sem f'' nem 0 egy kezdő x_0 pont és az x^* gyök között♦, továbbá az $f(x_0)*f''(x_0)>0$ teljesül a kezdőpontban♣, akkor az $\boldsymbol{f(x)=0}$ egyenlet gyökéhez konvergál az $\boldsymbol{x_{k+1}=x_k-f(x_k)/f'(x_k)}$. (Az állítás hátterében az igen általános és meglepően egyszerűen megfogalmazható Banach-Cacciopoli-Tyihonov-féle fixpont tétel áll. [γ/109])

2. A konvergencia *négyzetes* sebességű, azaz minden iterációs lépés megduplázza a helyes jegyek számát.

3. Az $\boldsymbol{f(x)=x^2-N}$ függvény teljesíti a fenti feltételeket; x_0-nak választható N is, ha N>1.

> Ugyanis:
>
> $f(x)=x^2-N=0 \Rightarrow \quad f'(x)=2x\neq 0 \Leftrightarrow x\neq 0$
>
> és $\quad f''(x)=2>0$
>
> és $\mathcal{L}$: $x_0=N \Rightarrow f(N)*f''(N)=(N^2-N)*2>0 \Leftrightarrow N^2>N \Leftrightarrow |N|>1$

Így gyökéhez (N=2 esetén $\sqrt{2}$-höz) fog (négyzetesen) konvergálni az

$$\boldsymbol{x_{k+1}=x_k-(x_k^2-N)/(2*x_k)\equiv \tfrac{1}{2}*(x_k+N/x_k) \rightarrow \sqrt{N}} \qquad \text{(NI-Racionális)}$$

Az N=2 esetén egyszerűbb alakhoz juthatunk: $x_{k+1}=x_k/2+1/x_k \rightarrow \sqrt{2}$.

♦ Ui. ha létezne olyan x, amelyre: $f'(x)=0$, akkor az érintő nem metszené az x-tengelyt, s így nem jelölné ki a következő x-t; ha pedig az $f''(x)=0$ (azaz az x inflexiós pont: domborúság↔homorúság váltó pont), akkor végtelenciklusra vezethet az iteráció.

♣ Így „néz” a gyök felé az érintő.

Az algoritmus, Pascal szintaxissal, az alábbi, amiben nyomon követjük a hatékonyságot jellemző iterációszámot is és a pontosságot (a helyes tizedes jegyek számát):

```
Procedure GyokN_Newton(Var N:Tort; Var vx:ValosSzam);
  Var
    y,z,gyok:Tort;
Begin
  iterszam:=0;
  { a kezdő, legalább 1 törtjegyre pontos közelítés megtalálása: }
  gyok:=N;
  Repeat
    { gyok:=(gyok+N/gyok)/2 }
    TortOszt(N,gyok,y); TortOsszead(gyok,y,z);
    TortOszt(z,KettoTort,gyok);
    Inc(iterszam);
    { pontosságszámítás:  y:=|gyök*gyök-N| }
    TortSzoroz(gyok,gyok,z);
    TortTavolsag(N,z,y);
    pont:=TortNagysagrend(y)+1;
  Until pont<=-1;
  { pontosítás: }
  Repeat
    { gyok:=(gyok+N/gyok)/2 }
    TortOszt(N,gyok,y); TortOsszead(gyok,y,z);
    TortOszt(z,KettoTort,gyok);
    Inc(iterszam); Inc(pont,pont);
  Until -pont>=MaxPontossag Div 2; [1]
  { output-konverzió:  gyok:Tört → vx:ValósSzám }
  Konv_RacVal(vx,gyok);
End; { GyokN_Newton }
```

Megfigyelésünket az alábbi táblázatban foglaltuk össze.

iterációszám	*a közelítés pontossága (tizedes jegyig)*	*a számítás ideje (másodperc)*
1.	***5***	***0.05 / 0.00***
2.	***11***	***0.05 / 0.00***
3.	***23***	***0.11 / 0.06***
4.	***48***	***0.17 / 0.12***
5.	***97***	***0.34 / 0.28***
8.	***195***	***0.83 / 0.77***

2.2. Pell-egyenlet alapján

[β/205-206]

A Newton-módszert jobban szemügyre véve, rájöhetünk, hogy elkerülhető a nagypontosságú racionális aritmetika, s elegendő „csak” a nagypontosságú egész-szám aritmetika. Tudván,

[1] E kilépési feltétel arra épít, hogy a helyes jegyek számát egy ciklusléps megkétszerezi.

hogy az iteráció során végülis racionális közelítő értékeken haladunk, x_k-ba helyettesítve P_k/Q_k törtet, (NI-Racionális) így alakul:

$$P_{k+1}/Q_{k+1}= ½(P_k/Q_k+N*Q_k/P_k)= (P_k^2+N*Q_k^2) / (2*P_k*Q_k) \qquad \text{(NI-Egész)}$$

Más szóval: a számítás egy racionális iteratív képlet kiszámolása helyett két egész-értékű iterációs transzformációvá szelídül:

$$\mathbf{P_{k+1}=P_k^2+N*Q_k^2} \qquad \textbf{és} \qquad \mathbf{Q_{k+1}=2*P_k*Q_k}$$

Pusztán ez a felismerés is gyorsíthatja a számítást, hiszen kikapcsolja a racionális aritmetika általános rutinjait. Ha emellett az ún. **Pell-egyenlet**et is kielégítő megoldások sorozatán keresztül jutunk a gyök közelébe, akkor tovább növeljük a számítás sebességét.

A módszer alapjairól:

1. Ha $N \in \mathbf{N}$ nem négyzetszám, akkor *végtelen sok* P-re és Q-ra ($P,Q \in \mathbf{N}$) teljesül az ún. **Pell-egyenlet**, vagyis a

   $$\mathbf{P^2–N*Q^2=4} \qquad \text{(Pell)}$$

   egyenlet.

2. A fenti P-k és Q-k hányadosa viszont konvergál a $\sqrt{N}$-hez.

Bizonyítás:

a) Végtelen sok (NI-Egész)-beli P_k, Q_k pár létezik, amely kielégíti a (Pell)-t is.

Tfh.: P_k, Q_k pár kielégíti a (Pell)-t!
(Ilyen biztosan van; pl.: N=2 esetén: 6, 4; egyéb N-re nem bizonyítjuk.)
Ekkor

$$\text{(Pell)} \Rightarrow N*Q_k^2=P_k^2–4 \qquad \text{(P2)}$$

No mármost találunk egy formulát, amellyel egy adott P, Q párból egy következő párt tudunk meghatározni, s így végső soron végtelen számút. Erre a célra alkalmas lesz az előbbi (NI-Egész):

$$P_{k+1}/Q_{k+1} = (P_k^2+N*Q_k^2) / (2*P_k*Q_k) =$$

$$[\text{(P2)} \Rightarrow] = (P_k^2+P_k^2–4) / (2*P_k*Q_k) =$$

$$= (2*P_k^2–4) / (2*P_k*Q_k) = (P_k^2–2) / (P_k*Q_k)$$

Így drasztikusan egyszerűsödik az iteráció:

$$\mathbf{P_{k+1}=P_k^2–2} \quad \textbf{és} \quad \mathbf{Q_{k+1}=P_k*Q_k} \qquad \text{(P3)}$$

Az így kapott P_{k+1}, Q_{k+1} pár is kielégíti a (Pell)-t. Mert:

$$P_{k+1}^2 - N*Q_{k+1}^2 = (P_k^2-2)^2 - N*(P_k*Q_k)^2 =$$
$$= P_k^4 - 4*P_k^2 + 4 - N*P_k^2*Q_k^2 =$$
$$= P_k^2*(P_k^2 - 4 - N*Q_k^2) + 4 =$$
$$[(P2) \Rightarrow] \quad = P_k^2*(N*Q_k^2 - N*Q_k^2) + 4 = 4$$

Vagyis beláttuk, létezik (legalább) egy (végtelen sok elemből álló, a Newton-iterációs formulát kielégítő) sorozat.

b) A fenti sorozat $\sqrt{N}$-hez való konvergenciája nyilvánvaló, hiszen (NI-Racionális)-beli a sorozat. Ennél többet is beláthatunk: a Pell-egyenletet kielégítő bármely sorozat is ugyanide konvergál:

Tfh.: P_k, Q_k pár kielégíti a (Pell)-t!

$$(Pell)/Q_k^2 \Rightarrow P_k^2/Q_k^2 - N = 4/Q_k^2$$

a jobb oldal 0-hoz konvergál, így

$$\mathbf{P_k^2/Q_k^2 \to N}$$

3. N=2 esetén a Pell-egyenletet kielégítő induló természetes számpár lehet a P_0=6726 és Q_0=4756 (sőt jobb a 39202 és a 27720). Ha $P_{k+1}=P_k^2-2$ és $Q_{k+1}=P_k*Q_k$ iterációval (l. (P3)) számoljuk a további P-t, Q-t, akkor $\mathbf{\lim P_k/Q_k = \sqrt{2}}$.

Az algoritmus:

```
Procedure Gyok2_Pell(Var vx: ValosSzam);
  Var
    p,q,y,z: EgeszSzam;
    racio  : Tort;
    {E: EgeszSzam, globális konstans.
     Értéke: E=1/ε, ahol ε a pontosság}
Begin
  Szovegbol_Egesz(p,'6726'); Szovegbol_Egesz(q,'4756');
  Szovegbol_Egesz(negy,'4'); Szoroz(negy,E,negyE);
  Repeat
    { q:=p*q; p:=p*p-2 }
    z:=q; Szoroz(p,z,q);
    Szoroz(p,p,y); Kivon(y,Ketto,p);
    { ciklusfeltétel: 4E<q*q kiértékelése }
    Szoroz(q,q,z);
  Until Nagyobb(z,negyE);
  Egeszbol_Tort(p,q,racio);
  Konv_RacVal(vx,racio);
End; { Gyok2_Pell }
```

A ciklusfeltétel levezetése:

ε a kívánt pontosság legyen 1/E alakú, ahol E egész szám, azaz $E=\frac{1}{\varepsilon}$. (E)

Addig kell a ciklusnak futnia, amíg: $\left|\left(\frac{p}{q}\right)^2 - 2\right| < \varepsilon$

Alakítsuk így: $\left|p^2 - 2q^2\right| < \varepsilon q^2$,

$$\begin{array}{ccc} & \uparrow & \uparrow \\ \text{(Pell)} \Rightarrow 4, & & \frac{1}{E}q^2 \Leftarrow \text{(E)} \end{array}$$

Azaz: $4E < q^2$

Mérésekkel is igazolható a gyorsaságnövekedés. Ennek tehát két oka van:

1. Maga a módszer elvileg is igen gyorsan konvergál, hisz lényege a Newton-módszer; annyival „erősebb” az eredeti Newton-módszernél, amennyivel „későbbi” tagtól indul, ill. amennyivel „olcsóbb” műveletekkel operál. (Az eredeti számítás 2 racionális osztást és összeadást tartalmaz, az utóbbi 2 szorzást és egy kivonást az egészek köréből.)

2. Az egész aritmetika rutinjait (a racionális közvetítése nélkül!) használja.

Elvileg indulhattunk volna más számpároktól is. Pl.: 6-4, 34-24, 198-140, 1154-816. Az utolsó még könnyen (azaz legfeljebb *LongInt* típusú adatként) számítható 6726-4756 utáni számpár a 39202-27720.

iterációszám	***a közelítés pontossága (tizedes jegyig)***	***a számítás ideje (másodperc)***
1.	***17***	***0.00***
2.	***35***	***0.00***
3.	***72***	***0.05***
4.	***146***	***0.11***

3. A π (Ludolph-szám)

3.1. A π racionális közelítései (3, 256/81, 157/50)

[α/21-25,87]

A Rhind-papirusz tanúsága szerint a babiloniak –akik annak a kornak legnagyobb matematikusait mondhatták magukénak– a kör kerületét helyettesítették a beírható hatszög kerületével; vagyis $K_6 = 6*r \equiv K_o$, miatt a π-t 3-nak tekintették. A matematikában kevesebb leleményt mutató egyiptomiak szerint –Imhotep óta– π az $4*(8/9)^2$-del, azaz 256/81 ≈ 3.16-tal egyenlő. (Imhotep hatszög helyett nyolcszöggel közelítette a kört.)

Arkhimédész a kört egy 96 oldalú szabályos sokszöggel közelítette. (A hatoldalúból kiindulva 4 ízben megduplázva az oldalszámot, kapja: 223/71<π<22/7.)

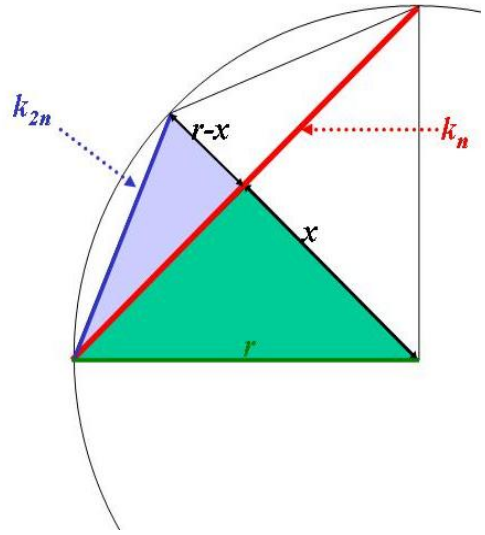
A kr.u.-i kínai matematikusok egyike –a 192 oldalú sokszöggel dolgozva– π-t 157/50-nek kapta, egy másik 355/113-nak, ami már a helyes érték első 6 tizedesét fölmutatta.

3.2. A kör közelítése szabályos sokszöggel

[β/210-211]

A módszer lényege annak fölismerése, hogy rekurzív összefüggés adható meg egy *n*-oldalú szabályos sokszög és egy *2n*-oldalú –ugyanazon körbeírt– sokszög kerülete között.[2] Jelöljük k_n-nel az *n*-oldalú sokszög egy oldalának hosszát. Ekkor igaz a következő rekurzió: $k_{2n} = \sqrt{2 - \sqrt{4 - k_n^2}}$ és pl. k_6=1 (hisz a szabályos 6-szög oldalhossza megegyezik a köré írt kör sugarával). Ennek a módszernek vált Ludolph van Ceulen is „áldozatává", aki végkimerülésig számolta egy 2^{62} oldalú sokszög alapján a π értékét, és 35 tizedes jegyét határozta meg (1610-ben).

[2] Az nyilvánvaló, hogy k_n-ből a K_o-t közelítő K_n számítható ($K_n = n*k_n$).



$k_n^2/4+x^2=r^2$ (1)

$(r-x)^2+k_n^2/4=k_{2n}^2$ (2)

(1)$\Rightarrow x^2=r^2-k_n^2/4$ (3)

(3)→(2) $(r-\sqrt{r^2-k_n^2/4})^2+k_n^2/4=k_{2n}^2$

$r^2-2\sqrt{r^2-k_n^2/4}+r^2-k_n^2/4+k_n^2/4=k_{2n}^2$

$2r^2-2\sqrt{r^2-k_n^2/4}=k_{2n}^2$

$2r^2-\sqrt{4r^2-k_n^2}=k_{2n}^2$ (4)

$\mathcal{L}$.: $r=1 \Rightarrow \lim n*k_n=2\pi$

s így (4)$\Rightarrow$ $2n*k_{2n}=2n*\sqrt{2-\sqrt{4-k_n^2}} \rightarrow 2\pi$

Az algoritmus alapja lehet a korábban ismertetett Newton-módszer sokszori, nagypontosságú számítása.

Megfigyelhető, hogy az igen sok irracionális művelet miatt kivárhatatlanul lassú, és emellett nagyon rossz az *elvi* konvergenciasebesség szempontjából is.

3.3. Wallis „racionális szorzat formulája”

[β/211]

A módszert pontosan megadja az alábbi formula:

$$\frac{\pi}{2}=\frac{2*2*4*4*6*6*...}{1*3*3*5*5*7*...}$$

Az algoritmus:

```
Procedure Pi_Wallis(Var vx : ValosSzam);
  Var
    sz,ne : EgeszSzam;
    pi,x,y: Tort;
    paros : Boolean;
Begin
    { pi:=2; sz:=2; ne:=1; paros:=False }
  pi:=KettoTort; sz:=Ketto; ne:=Egy; paros:=False;
  Repeat
    Egeszbol_Tort(sz,ne,x); TortSzoroz(pi,x,y); pi:=y;
    If paros then begin Novel(sz); Novel(sz) end
            else begin Novel(ne); Novel(ne) end;
    paros:=not paros;
  Until {pi.szamlalo.hossz=Pontossag-1} Kerekites;
  Konv_RacVal(vx,pi);
End; { Pi_Wallis }
```

Megfigyelhető, hogy rendkívül rosszul konvergál. Ha a számítás során pl. egy 100 jegyű számlálójú racionális számhoz jutunk, a π közelítésnek még csak az első tizedes jegy lesz pontos.

A számítás ideje: ...

iterációszám	*a közelítés pontossága (tizedes jegyig)*	*a számítás ideje (másodperc)*
1.		
2.		
3.		
4.		

3.4. Az arctg hatványsoraira épülő módszerek

[α/136,β/212-214]

A módszerről:

A döntő fordulat James Gregory nevéhez fűződik, aki 1671-ben fölfedezte *arctg*-függvény hatványsorát:

$$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \qquad x \le 1 \qquad \text{(Arc)}$$

Erre és a következő „addíciós” formulára épülnek a későbbi, π-t közelítő algoritmusok:

$$\operatorname{arctg} x + \operatorname{arctg} y = \operatorname{arctg} (x+y)/(1-xy). \qquad \text{(ArcAdd)}$$

Az addíciós formula szerepe a következő: mivel az arctg fenti (Arc) hatványsora annál jobban konvergál, minél kisebb az argumentuma, célszerű kicsi értékek arcus tangensét számoltatni. Így e formulával „gyárthatunk” több, kisebb argumentumú összegeket, azaz egy lassan konvergáló összeget kicserélünk több, de gyorsan célt érő összeggel. Például: arctg 1 = arctg ½ + arctg ⅓.

> Ugyanis az
>
> $$\operatorname{arctg} 1 = \operatorname{arctg} x + \operatorname{arctg} y$$
>
> azonossághoz kerestünk alkalmas x, y-t.
>
> Az (ArcAdd) formula jobboldala szerint arctg 1 = arctg (x+y)/(1-xy), azaz az
>
> $$1 = (x+y)/(1-xy)$$
>
> formulából fejezzük ki y-t:
>
> $$y = (1-x)/(1+x).$$
>
> Behelyettesítve x-be ½-t, y-ra kijön ⅓.

Az algoritmusok:

Az arcus tangens kiszámítása a fenti (Arc) hatványsor alapján így történhet:

```
Procedure ArcTg(Var x,atx: Tort);
                { arcTg(x) = x-x^3/3+x^5/5-...x^n/n... }
  Var
    n     : EgeszSzam;
    m,m2,
    mm,nt : Tort;
    paros : Boolean;
Begin
  { atx:=x (arctg x közelítő értéke);
    m:=x (az aktuális tag a hatványsorban);
    m2:=x^2 (az egymást követő tagok szorzótényezője) }
  paros:=False; atx:=x; m:=x; TortSzoroz(x,x,m2);
  n:=Egy;
  Repeat
    Novel(n); Novel(n);
    Egeszbol_Tort(Egy,n,nt); {köv. nevező}
    TortSzoroz(m,m2,mm);     {köv. számláló}
    TortSzoroz(mm,nt,m);     {köv. tag}
    paros:=not paros;
    If paros then Begin TortKivon(atx,m,atx) End
            else Begin TortOsszead(atx,m,atx) End;
  Until atx.nevezo.hossz>=Pontossag-5 {Kerekites};
End; { ArcTg }
```

Láthatóan kedvezőtlen lassúsággal konvergál (különösen nagyobb argumentumok esetén). Ennél hatékonyabb, ha egy ciklusléptében a két egymást követő tagot számítunk ki, mivel az így „átdefiniált” sor szaporább konvergenciájú. (Sőt kedvezőbb algoritmikailag is.)

```
Procedure ArcTg2(Var x,atx: Tort);  { arcTg(x)=
                        =(x/1-x^3/3)+(x^5/5-x^7/7)+... }
  Var
    n     : EgeszSzam;
    m,mm,
    x2,x4,
    nt,nt2 : Tort;
Begin
  { atx:=x (arctgx közelítő értéke);
    x2:=x*x; x4:=x^4 (az egymást követő tagok
    szorzótényezője) }
  TortSzoroz(x,x,x2); TortSzoroz(x2,x2,x4);
  n:=Egy; atx:=NullaTort;
  Repeat
    Egeszbol_Tort(Egy,n,nt);
    Novel(n); Novel(n);
    Egeszbol_Tort(Egy,n,nt2);
    Novel(n); Novel(n);
    TortSzoroz(x2,nt2,m); TortKivon(nt,m,mm);
    TortSzoroz(mm,x,m);
    TortOsszead(atx,m,atx);
    TortSzoroz(x,x4,m); x:=m;
  Until MaxHossz(atx)>=MaxPontossag-5;
  Becsul(atx,MaxPontossag-50);
End; { ArcTg2 }
```

Az első π-számítást az ingerlően egyszerű összefüggés alapján végezzük (Leibniz-sor):

$$\pi/4 = \text{arctg } 1 .$$

```
Procedure Pi_Arctg0(Var vx : ValosSzam); { pi=
                                          =4*arctg(1) }
  Var
    pi1,pi2 : Tort;
Begin
  ArcTg(EgyTort,pi1);
  TortSzoroz(pi1,NegyTort,pi2);
  Konv_RacVal(vx,pi2);
End; { Pi_Arctg0 }
```

Kellemetlen tapasztalatunk, hogy igen sok számolás után is nagyon gyengén közelítő értékhez jutunk. Az arctg-nek 100 tagját kiszámolva is csak az első jegy lesz helyes!

A következő módszer 1794-ből, Georg von Vegatól származik:

$$\pi/4 = 5 \text{ arctg } (1/7) + 2 \text{ arctg } (3/79).$$

```
Procedure Pi_Arctg1(Var vx : ValosSzam); { pi=
                              20*arctg(1/7)+8*arctg(3/79) }
  Var
    pi1,pi2,pi3 : Tort;
Begin
  ArcTg(Heted,pi1);
  TortSzoroz(pi1,HuszTort,pi2);
  ArcTg(HaromHetvenkilenced,pi1);
  TortSzoroz(pi1,NyolcTort,pi3);
  TortOsszead(pi2,pi3,pi1);
  Konv_RacVal(vx,pi1);
End; { Pi_Arctg1 }
```

Egy másik megközelítése ugyanennek a

$$\pi = 24*\text{arctg}(1/8)+8*\text{arctg}(1/57)+4*\text{arctg}(1/239)$$

formula alapján.

```
Procedure Pi_Arctg2(Var vx : ValosSzam); { pi=
             24*arctg(1/8)+8*arctg(1/57)+4*arctg(1/239) }
  Var
    pi1,pi2,pi3 : Tort;
Begin
  ArcTg(Nyolcad,pi1);
  TortSzoroz(pi1,HuszonNegyTort,pi2);
  ArcTg(OtvenHeted,pi1);
  TortSzoroz(pi1,NyolcTort,pi3);
  TortOsszead(pi2,pi3,pi1);
  ArcTg(KetszazHarmincKilenced,pi2);
  TortSzoroz(pi2,NegyTort,pi3);
  TortOsszead(pi1,pi3,pi2);
  Konv_RacVal(vx,pi2);
End; { Pi_Arctg2 }
```

Megfigyelhető, hogy

1. nem ugyanannyiszoros iterációval számítja az egyes tagokat, ami eleve fölveti a pontosság kérdését;

2. valószínűleg a konvergencia is gyorsítható (a konzisztencia növelése mellett) azzal, ha egyetlen számítási ciklusba sűrítjük a π kiszámítását.

A számítás ideje: ...

iterációszám	***a közelítés pontossága (tizedes jegyig)***	***a számítás ideje (másodperc)***
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

4. Az e

Az *e* történetéből: ...

4.1. Az $(1+1/n)^n$ „határértéke”

Az $(1+1/n)^n$ fokozatos számításáról:

Célszerű n-t 2-hatványnak választani, mivel így egy iterációs lépéssel exponenciálisan növekvő pontosságú közelítő értékhez jutunk.

1. $1+1/2^k \rightarrow 1+1/2^{k+1}$

```
n=2^k,          akkor     2^(k+1)=2*n;
x:=1+1/n,  akkor     1+1/2^(k+1)=(1+x)/2
```

2. $(1+1/n)^n \rightarrow (1+1/(2*n))^{2*n}$

```
x:=(1+x)/2; z:=x
Ciklus 2*n-szer
  z:=z*z
Ciklus vége
x:=z
```

Az algoritmus:

```
Procedure E_hatvany(Var vx : ValosSzam);
  Var
    x,y,z : Tort;    k,kitevo: Integer;
Begin
  { x:=(1+1/n)^n  n=2^k }
  TortOsszead(EgyTort,Egytort,KettoTort);
  x:=KettoTort {x=(1+1/n)};  kitevo:=0;
  Repeat
    { n:=2*n; x:=(1+1/n)^n  n=2^k <=>  x:=(1+x)/2 }
    Inc(kitevo);
    { x:=(1+1/n)=(1+x)/2 }
    TortOsszead(x,EgyTort,y); TortOszt(y,KettoTort,x);
    z:=x;
    For k:=1 to kitevo do
    Begin
      TortSzoroz(z,z,y);  z:=y
    End;
  Until (z.szamlalo.hossz+z.nevezo.hossz)>=Pontossag;
  Konv_RacVal(vx,z);
End; { E_hatvany }
```

Megfigyelhető, hogy kellemetlenül lassú a konvergencia.[3] Amire közelítő racionális szám számláló jegyeinek száma (és természetesen a nevezőé is) eléri a lehetséges maximumot, addig a hányadosnak csak kevés jegye lesz pontos. Pl.: a 6. iterációs lépésre a számláló „hossza” meghaladja a 100-t, de a közelítése így kezdődik: 2.69... .

A számítás ideje: ...

iterációszám	*a közelítés pontossága (tizedes jegyig)*	*a számítás ideje (másodperc)*
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

4.2. Az *e* Taylor-sorára épülő módszerek

[β/208]

A módszerről:

$$e^x := \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

A konvergenciáról jó jellemzést ad pl. a Lagrange-maradéktagos alakja:

$$e^x := \sum_{k=0}^{n} \frac{x^k}{k!} \quad + \quad \frac{e^{\xi}}{(n+1)!} x^{n+1}$$

(A $\xi<x$. Így x=1 esetén a hibatag O(1/n!) nagyságrendben tart a 0-hoz.)

Az algoritmus:

```
Procedure E_Taylor(Var vx : ValosSzam);
  Var
    fakt,e : array [false..true] of Tort;
    nt     : Tort;
    n      : EgeszSzam;
    paros  : Boolean;
Begin
  iterszam:=0;
  { n:=2;  e:=2+1/2;  fakt:=1/2! }
  { a ide-oda másolások elkerülésére duplapufferelve }
```

[3] Hiszen, minden egyes közelítő érték kiszámításához sok-sok szorzásra van szükség. (Gondoljon bele pl. a 100 000-dik tag kiszámítása milyen nagyságrendű időt igényel.

```
    paros:=True; n:=Ketto; fakt[paros]:=Fel;
    TortOsszead(KettoTort,Fel,e[paros]);
    Repeat
      { n:=n+1;  fakt:=fakt/n;  e:=e+fakt }
      Novel(n);  Egeszbol_tort(n,Egy,nt);
      TortOszt(fakt[paros],nt,fakt[not paros]);
      TortOsszead(e[paros],fakt[not paros],e[not paros]);
      paros:=not paros;
      Inc(iterszam);
    Until (e[paros].szamlalo.hossz+e[paros].nevezo.hossz)>=
                                                MaxPontossag;
    {gyakorlatilag annyi jegyre pontos,ahány jegyű a
     nevező}
    Konv_RacVal(vx,e[paros],MaxPontossag);
  End; { E_Taylor }
```

Megfigyelhető, hogy körülbelül a közelítő tört számláló (és nevező) számjegyeivel azonos számú jegye lesz a pontos értékkel megegyező!

5. Nagy prímek Lucas-próba alapján

[β/ 222-223]

A módszerről:

Az algoritmus:

Megfigyelhető, hogy

Függelék

Hivatkozott irodalom

α. Forica T. Cimpan: A π története
β. J.Nievergelt-J.C.Farrar-E.M.Reingold: Matematikai problémák megoldásának számítógépes módszerei
γ. Szidarovszky F.: Bevezetés a numerikus módszerekbe
δ. B.L.van der Waerden: Egy tudomány ébredése
ε. Sain M.: Nincs királyi út

A szereplőkről

Imhotep (kr.e. ~1840)
Pithagorasz (kr.e. 560-480)
Arisztotelész (kr.e. 384-322)
Euklidész (kr.e. 365-300 vagy 330-275)
Arkhimedész (kr.e. 287-212)
Ludolph (1540-1610)
Napier (1550-1617.04.04)
Mersenne (1588.09.08-1648.09.01)
Pell (1611.03.01-1685.12.12)
Wallis (1616.12.03-1703.11.08)
Gregory (1638.11. -1675.10.)
Leibniz (1646.07.01-1716.11.14)
Sharp (1651-1742) - π 72 jegy
Euler (1707.04.15-1783.09.18) - 1727: e-jelölés, 23 jegy

Vázlat: