EGYVÁLTOZÓS FÜGGVÉNYEK ÁBRÁZOLÁSA

Szlávi Péter

2000-2012

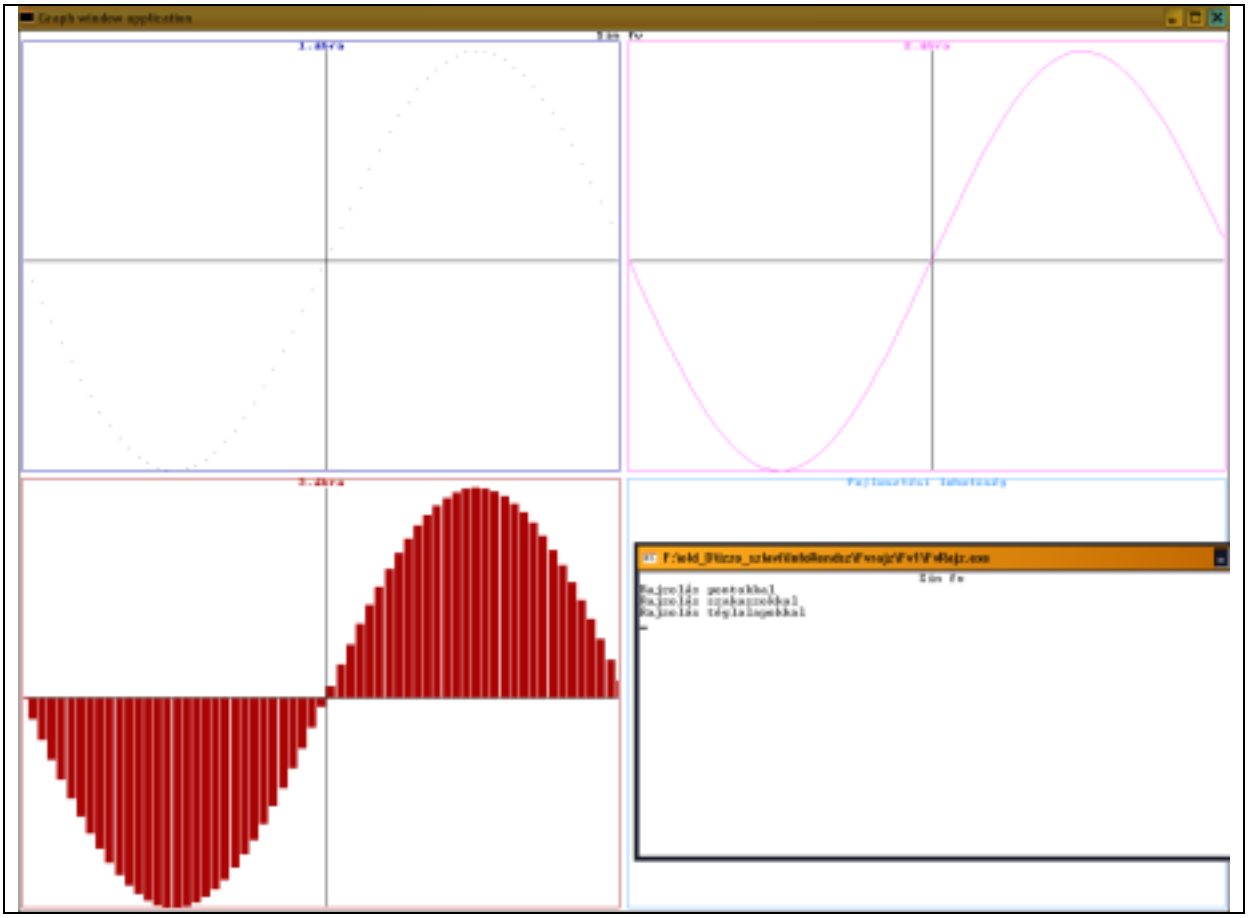
TARTALOM

1 Bevezetés

Feladat egy egyváltozós valós függvény kirajzolása különféle megjelenítési módszerekkel. Például:

- pontokkal,
- folytonos szakaszokkal,
- téglalapokkal,
- … .

Az elvárásokat legjobban az alábbi, futás során keletkezett ábrasor fejezi ki,



1. ábra. Egy futási kép – a sinus függvény.
(FP-ben két ablak: az egyik a vezérléshez, a másik a grafikus megjelenítéshez kell.)

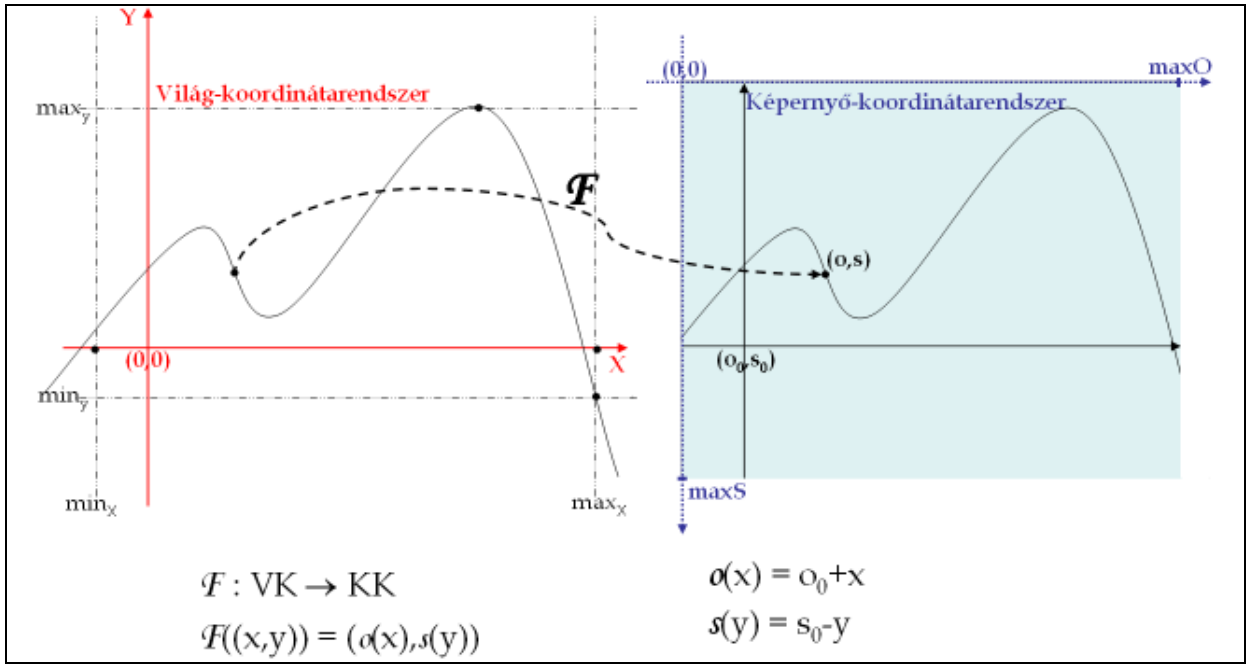
valamint egy (teljesebb) próba: Fv1Rajz.exe.

2 Útban a megoldás felé

2.1 Jelölések

Alapadatok:

- f – függvénykapcsolat; f: $D_f \rightarrow R_f$,
- D_f – (az „érdekes”) értelmezési tartomány = $[\min_x..\max_x]$
- R_f – (az „érdekes”) értékkészlet = $[\min_y..\max_y]$



2. ábra. A transzformáció „ránézésre”.
(VK = világ-koordinátarendszer, KK = képernyő-koordinátarendszer)

A (lineáris) transzformáció ellenőrzése 2, különböző pontra:

- Az *origó* leképezése: $F((0,0)) = (o_0+0, s_0-0) = (o_0, s_0)$
- A *bal-felső sarok* leképezése: $F((-o_0, s_0)) = (o_0-o_0, s_0-s_0) = (0,0)$

2.2 Problémák

1. túl szűk/tág az értékkészlete
2. túl szűk/tág az értelmezési tartománya
3. függőlegesen alul/felül kilóg
4. vízszintesen balra/jobbra kilóg

2.3 A problémák megoldása

Léptékezés (1-2.-re), az origó eltolása (3-4.-re).

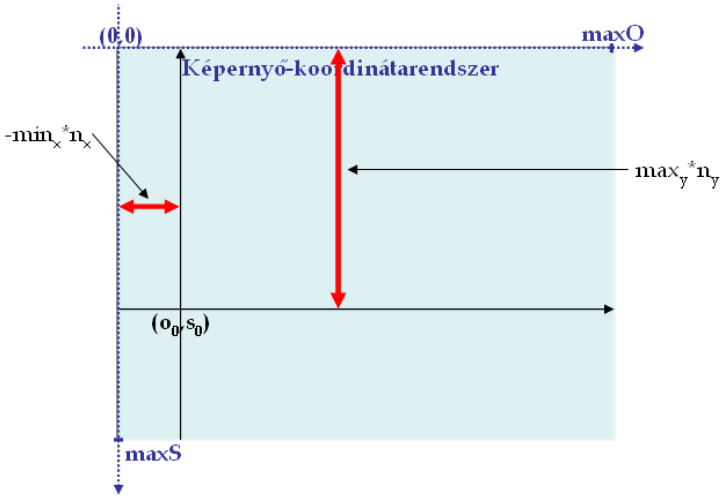
Nyújtás (a teljes [0..maxO]×[0..maxS] képernyőre)

- X-irányban: $n_x = (maxO+1) / (max_x-min_x)$
- Y-irányban: $n_y = (maxS+1) / (max_y-min_y)$

A számlálóbeli +1 a képernyő-koordinátarendszer egész léptékének köszönhető: annyi darab pont van az oszlopokban, illetve a sorokban.

Origóeltolás

- $o_0 = -min_x * n_x$
- $s_0 = max_y * n_y$
  $\equiv maxS + min_y * n_y$



Így a keresett **F leképezés**:

- $o(x) = o_0 + x * n_x$
- $s(y) = s_0 - y * n_y$

3 Módszerek – algoritmusok

Az algoritmus ötlete pofonegyszerű:

1. generáljuk le az ábrázolandó függvényt egy ún. függvénytáblába, valamekkora lépésközzel, célszerűen: ekvidisztáns alappontok felett, növekvő x-ek mentén;
2. majd (most már függvény kiszámítási szabályától függetlenül) rajzoljuk ki a függvénytáblában lévő pontokon nyugvó függvénygörbét, valamilyen módszerrel.

Algoritmikus adatok és egyéb kellékek:

```
Függvény f(Konstans x:Valós):Valós
  … a kirajzolandó függvényt leíró függvényeljárás …

Típus
  TKoordináták = Tömb(1..MaxN:Valós) [koordináták]
  TFüggvénytábla = Rekord(
                     n:Egész,
                     x,y:TKoordináták [1],
                     minY,maxY:Valós)

Változó [globálisak]
  ft:TFüggvénytábla                [egy fv. gráfjának pontjai]
  mag,szél:Egész                   [az aktuális képernyő ablak méretei]

Változó [az egyes módszerekhez, lokálisak]
  o0,s0:Egész                      [az origó helye a képernyőn]
  nx,ny:Valós                      [nyújtási tényezők]
  dx:Egész                         [oszlopszélesség]
  eo,es:Egész                      [az előzőleg kirajzolt pont]

Eljárás PontRajzolás(Konstans x,y:Valós):
  [Ef: helyes o0,s0, nx,ny]
  Változó
    o,s:Egész
  o:=o0+Kerekít(x*nx); s:=s0-Kerekít(y*ny)
  Pont(o,s)
  [eo:=o; es:=s] [2]
Eljárás vége.
```

[1] Az x-koordináták növekvően rendezettek ⇒ `minX=ft.x(1), maxX=ft.x(ft.n)`.

[2] A `SzakaszRajzolás` eljárás majd épít arra, hogy az utoljára kirajzolt pont helye meglegyen az `(eo,es)`-ben.

```
[A Pont eljárás ún. grafikus primitív, Turbo Grafikában [3]: PutPixel.]
Eljárás SzakaszRajzolás(Konstans hozX,hozY:Valós):
  [Ef: az (eo,es) pont már ki van rajzolva, s ez az aktuális]
  Változó
    s,o:Egész
  o:=o0+Kerekít(hozX*nx); s:=s0-Kerekít(hozY*ny)
  Szakasz(eo,es,o,s)
  eo:=o; es:=s
Eljárás vége.
[A Szakasz eljárás ún. grafikus primitív, TG-ben: Line.]
Eljárás TéglaRajzolás(Konstans x,y:Valós):
  [Ef: dx=az oszlopok szélessége (Egész)]
  Változó
    s,o:Egész
  o:=o0+Kerekít(x*nx); s:=s0-Kerekít(y*ny)
  Tégla(o,s,o+dx,o0-sgn(y) [ne érjen rá az x-tengelyre! [4]] )
Eljárás vége.
[A Tégla eljárás ún. grafikus primitív, TG-ben: Bar.]
```

3.1 Bamba, de egyszerű…

Nem figyeljük a „képernyőre miként kerülést”… az origó középen, léptékezés nincs. A „rá-nézésre” transzformáció méltó párja:

```
…
o0:=maxO Div 2; s0:=maxS Div 2
Ciklus i=1-től ft.n-ig
  PontRajzolás(ft.x(i),ft.y(i))
Ciklus vége
…
```

3.2 Képernyőre normálva

```
[ef: ft.x szigorúan növekvően rendezett tömb ⇒
          ∀i∈(1..ft.n): ft.x(1)<ft.x(i)<ft.x(ft.n)]

…
nx:=(maxO+1)/(ft.x(ft.n)-ft.x(1)); ny:=(maxS+1)/(ft.maxY-ft.minY)
o0:=-nx*ft.x(1); s0:=maxS+ny*minY
Ciklus i=1-től ft.n-ig
  PontRajzolás(ft.x(i),ft.y(i))
Ciklus vége
…
```

[3] Akár Turbo Pascalban, akár Free Pascalban a `Graph` unit tartalmazza a grafikus „szókincset”.
[4] Meggondolandó az `x=0`, és az `y=0` esete!

3.3 Aránytartva normálva

```
[ef: ft.x szigorúan növekvően rendezett tömb]

…
nx:=(maxO+1)/(ft.x(ft.n)-ft.x(1)); ny:=(maxS+1)/(ft.maxY-ft.minY)
o0:=-nx*ft.x(1); s0:=maxS+ny*minY
Ha ny>nx akkor ny:=nx különben nx:=ny [egyszerűbben: nx,ny:=Min(nx,ny)]
Ciklus i=1-től ft.n-ig
  PontRajzolás(ft.x(i),ft.y(i))
Ciklus vége
…
```

Hogyan lehetne megvalósítani egy ábrán több függvény kirajzolását, ha ügyelni kell az arányokra (az egymáshoz s esetleg: saját magukhoz)? Fölteheti, hogy a függvények értelmezési tartománya ugyanaz, sőt az alappontok is megegyeznek.

3.4 Folytonos vonallal

```
[ef: ft.x szigorúan növekvően rendezett tömb]

… [a 3.2 vagy a 3.3 inicializálása] …
PontRajzolás(ft.x(1),ft.y(1))
Ciklus i=2-től ft.n-ig
  SzakaszRajzolás(fv.x(i),ft.y(i))
Ciklus vége
…
```

3.5 Téglákkal

```
[ef: ft.x szigorúan növekvően rendezett, és azonos különbségű tömb]

… [a 3.2 vagy a 3.3 inicializálása] …
Ciklus i=1-től ft.n-ig
  TéglaRajzolás(ft.x(i),ft.y(i))
Ciklus vége
…
```

3.6 Trapézokkal

Ez ötvözi a szakaszokkal és a téglákkal történő rajzolást. Hátránya, általában kevesebb segítséget nyújtanak a programnyelvek. [5] Elemibb műveletekkel (szakaszrajzolás és tartományszínezés) persze nem nagy munka árán megoldható.

Az algoritmizálás és kódolás: hf.

3.7 Görbeívekkel

Számos módszer közül választhatunk. Legkézenfekvőbb, hogy az N pontra ráfektethetünk egy N-1-ed fokú polinomot. Másik szokásos módszer (ill. inkább módszercsalád), hogy a szomszédos pontokra valahanyadfokú (de egyedenként paraméterezett) polinomot illesztünk úgy, hogy az adott pontokon átmenjenek, sőt –hogy ez az „átmenet” az egyik polinomból a másikba minél „simább” legyen– az érintők egyenlőségét is meg szokták követelni.

Az algoritmizálás és kódolás: hf.

---

[5] Üdítő kivétel a Turbo Grafika e célra kiváló két eljárása:

```
Procedure DrawPoly(Const db:Word; Const pontok);
Procedure FillPoly(Const db:Word; Const pontok);
{a pontok: db darab a Graph unitban definiált PointType típusú adat kezdőcíme}
Type PointType=Record x,y:Integer End;
```

Segítségképpen alább egy könnyen megvalósítható módszerről ötletelek:

1. Az 1-3 pontra ráfektetünk egy parabolaívet. (Ez egyértelműen elvégezhető.)
2. A 3.-tól az egymást követő pontpárokra úgy fektetünk parabolaíveket, hogy az a bal oldali szomszédjához „érintőlegesen” is illeszkedjen.
3. A parabolaíveket $f_i(x)=a_ix^2+b_ix+c$ alakban írjuk föl. Az A**x**=**b** vektoregyenletet kell megoldanunk, úgy hogy az A együtthatómátrixot tartalmazza azon lineáris egyenletek alappontok x-eiből számított konstansait, amelyek az 1-2. alapján készültek. Az A**x**=**b** egyenlet túl oldali vektorának elemei (**b**) részben az y-okból fog állni, részben egyéb megfontolásokból származnak.
4. Képezzük az együtthatómátrix inverzét (A^{-1}).
5. Megoldjuk az egyenletet: **x**=A^{-1}**b**.

Megjegyzem: van számos hatékony módszer a numerikus matematikában lineáris egyenletrendszerek megoldására; így a mátrix invertálás valójában nem kardinális. Ilyen módszer például a Gauss-Jordan elimináció. (Ez az **x** vektor mellett az A^{-1}-et is előállítja, bár minket csak az **x** érdekel.) [6]

Nézzünk egy konkrét példát!

Ha 4 pontra kell fektetni görbeíveket, akkor 2 parabolaívvel oldható meg a probléma. Az első 3 pontra egyértelműen fektethetünk egyet, majd a 3. és 4. pontra azzal a plusz feltétellel, hogy ez utóbbi érintője a 3. pontban egyezzen meg az első ugyanezen pontbeli érintőjével. Azaz a keresett 6 ismeretlen paraméterre az alábbi egyenleteket állíthatjuk föl.

$f_1(x_1)=y_1$, $f_1(x_2)=y_2$, $f_1(x_3)=y_3$
$f_2(x_3)=y_3$, $f_2(x_4)=y_4$, $f_1'(x_3)-f_2'(x_3)=0$

A részletszámítások nélkülözésével, csak az egyes lépések végeredményét mutatja az alábbi ábra.

i:	1	2	3	4
x_i:	0	1	2	3
y_i:	1	4	1	4

b	Mátrix (A):						Egyenletek:
							$f_i(x)=a_ix^2+b_ix+c_i$
1	0	0	1	0	0	0	← $f_1(x_1)=y_1$
4	1	1	1	0	0	0	← $f_1(x_2)=y_2$
1	4	2	1	0	0	0	← $f_1(x_3)=y_3$
1	0	0	0	4	2	1	← $f_2(x_3)=y_3$
4	0	0	0	9	3	1	← $f_2(x_4)=y_4$
0	4	1	0	-4	-1	0	← $f_1'(x_3)-f_2'(x_3)=0$
	a_1	b_1	c_1	a_2	b_2	c_2	
x	Inverz mátrix (A^{-1}):						
-3	1	-1	1	0	0	0	
6	-2	2	-1	0	0	0	
1	1	0	0	0	0	0	
9	-1	2	-2	-1	1	1	
-42	3	-10	8	4	-4	-5	
49	-3	12	-9	-3	4	6	

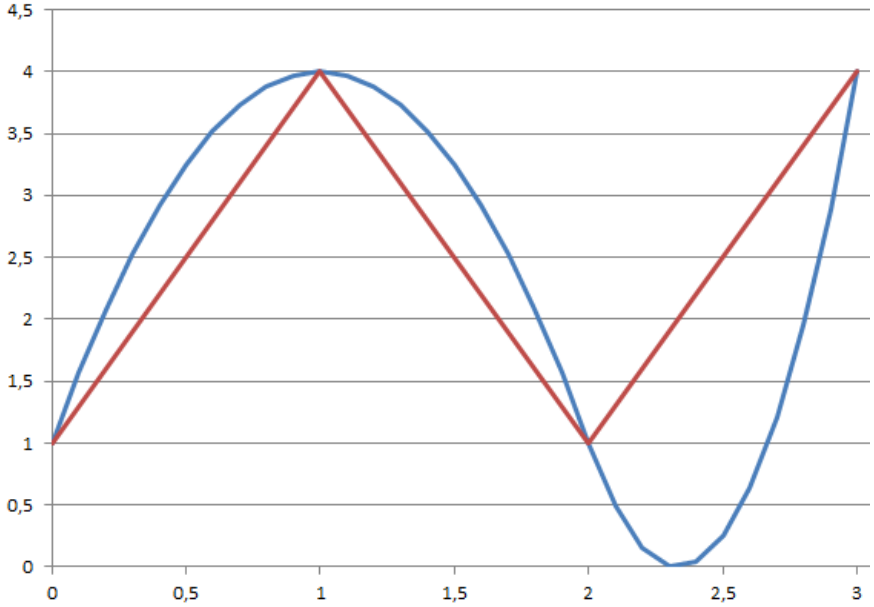
Tehát megkaptuk a két parabolaív paramétereit:

$f_1(x)=-3x^2+6x+1 \quad x\in[x_1..x_3]$, $\quad f_2(x)=9x^2-42x+49 \quad x\in[x_3..x_4]$

Hogy érjünk is ezzel az izzasztó eredménnyel valamit, az adott x-pontok között finomabb lépésekben is számíthatjuk az interpolált értékeket. Ezáltal kevés pont esetén is „szép” görbével tudjuk megjeleníteni az adott görbeívet.

[6] Ennek kipróbálására szíves figyelmükbe ajánlom:
GaussJordan.exe programot, illetve a forrását: GaussJordan.pas!

Lássuk, mit kapunk a kirajzolás után a fenti négy pontra illeszkedő függvénygörbe gyanánt!



Ahhoz képest, hogy eredetileg összesen csak négy pont volt adva (pirossal szakaszokkal jelölve), elégedettek lehetünk a látvánnyal (kékkel jelölve). Itt összesen 8 köztes pontot számoltunk az egyes rögzített pontok között.

Az itt körvonalazott módszer (kirajzolás nélküli, a keretbe egy az egyben nem biellisztheto) megvalósítását is megtalálhatja: GaussJorda.zip.

4 A keretprogram

Lásd `Fv1RajzK.htm`, `Fv1RajzK.pas`.

5 A megoldás

A 3 tervezett és fentebb kidolgozott megoldást tartalmazzák az alábbi fájlok: `Fv1Rajz.htm`, `Fv1Rajz.pas`.

Házi feladatként érdemes a 3.6-ban vagy a 3.7-ben körvonalazott megjelenítési módokat megvalósítani. A 3.6-hoz lényegileg minden ismert (az anyag alapján). A 3.7-ben körvonalazott módszer (a hivatkozott programban megvalósított kóddarab felhasználásával) már viszonylag könnyen elkészíthető. „Csupán” az adott szomszédos pontokat összekötő parabolaíveket finomabb lépésközű „alpontokra” kell illeszteni, (pontokkal, szakaszokkal vagy téglákkal) kirajzolni.