[1] W.H. Press et al. : Numerical Recipies, Cambridge University Press , 1994.

(C, Fortran, Pascal)

Fortran stílusú programok

60-as évek algoritmusai (néhány újabb is akad)

Ízelítő, ha létfontosságú, vannak rafináltabb módszerek is (?? www)

Bizonyítások, levezetések csak elvétve

Inkább mindenről általános ismertető, úgyis gyakorlatban kell csinálni

Interpoláció és extrapoláció egy és több dimenzióban

A probléma

Ismert:

$$x_1, x_2, ...x_N$$
$$y_1 = f(x_1), y_2 = f(x_2), ...y_N = f(x_N),$$

de f analitikusan nem ismert. Pl. mérés, szimuláció eredménye. Az x_i -k közötti távolság lehet egyforma, vagy különböző.

Cél:

$f(x)$meghatározása tetszőleges x esetén.

$x_1 \leq x \leq x_n$ interpoláció
egyébként extrapoláció

Példák: extrapoláció: jóslások (időjárás, tőzsde), differenciál egyenlet megoldásnál véges lépéshosszal való eredményből extrapolálhatunk a 0 lépéshosszra

nem egyenletes mintavételezésről egyenletesre áttérés

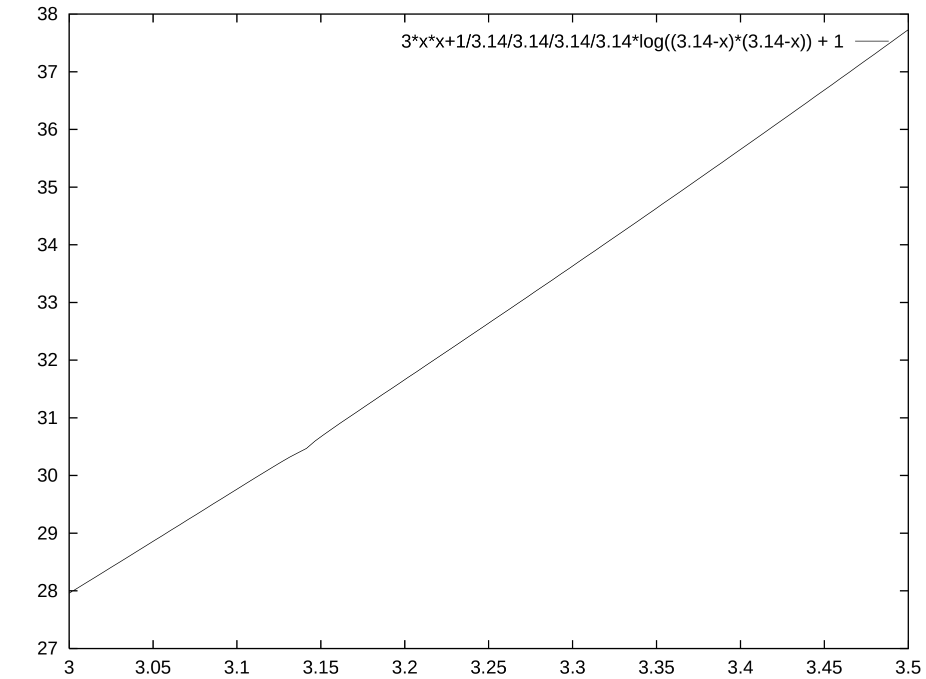
2 dimenziós illesztés: digitális képek nagyítása, átskálázása

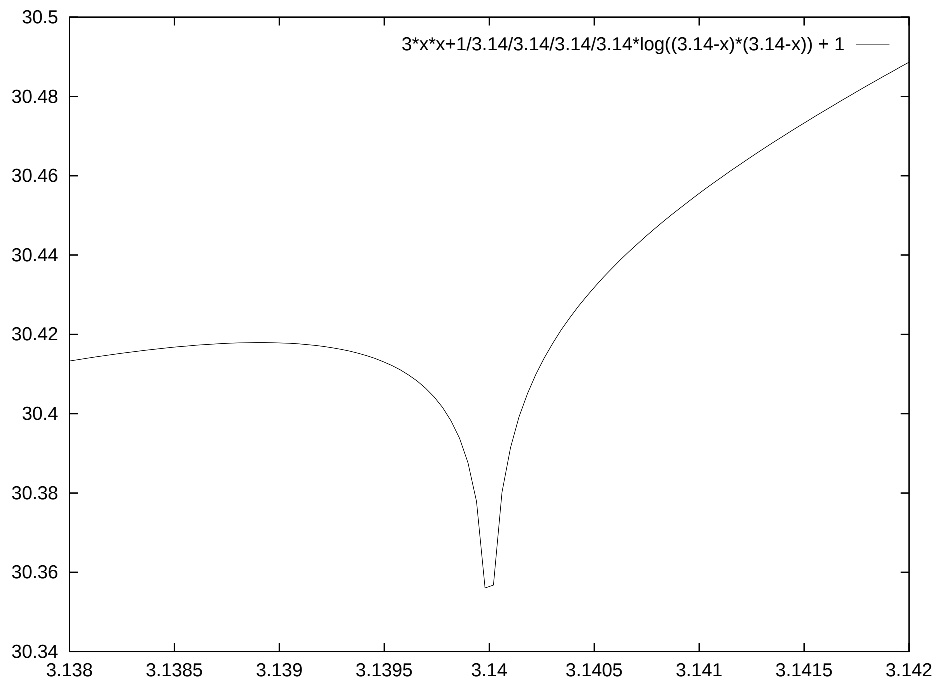
Feltételezés: (általában)

f:	folytonos	
	sima	(deriváltak folytonosak)

Természetesen vannak nehezen, vagy egyáltalán nem interpolálható függvények. Egy érdekes analitikus példa:

$$f(x) = 3x^2 + 1/\pi^4 ln[(\pi - x)^2] + 1$$





Az ismert pontok között f -et *modellezzük* valamilyen általános függvényalakkal. Ez a függvény függhet a konkrét problémától.

Leggyakoribb alakok:

polinom
racionális függvény (polinomok hányadosa)
trigonometrikus függvény, stb.

Megj: *A függvény illesztés, közelítés más probléma. Ott ismert globális függvényalakot illesztünk a mérési hibákkal terhelt adatokra. Itt az interpolált függvény egzaktul átmegy az ismert pontokon.*

A megoldás általános vázlata

Ha sok ismeretlen x helyen meg kell határozni $f(x)$ értékét, illetve kellenek pl. a polinom koefficiensei:

1. illesztünk $x_1, x_2, ...x_N$ -re

2. kiszámoljuk $f(x)$ -et

Probléma lehet:

numerikus kerekítések miatti pontatlanság
nem elég hatékony numerikusan

Kevesebb ismeretlen pont, vagy a numerikus pontosság fontos:

1. $f(x)$ közelítése a legközelebbi néhány (k) $\{x_i, f(x_i)\}$ alapján
2. $k \to N$ remélhetőleg közben $f(x)$ konvergál. Az utolsó lépésbeli változás lehet az interpoláció *hibá*jának becslése.

Az interpoláció lehet globális vagy lokális.

- Globális: az összes pontra illesztünk pl. $N - 1$ -ed rendű polimomot. (2 ponton át mindig húzható egyenes, 3-n parabola, stb.)

- Lokális: felosztjuk az ismert pontokat szomszédos pont n -esekre, és erre illesztünk $n - 1$ -ed rendű polinom-szakaszokat

Az interpoláció *rendje*: az egy illesztésnél felhasznált pontok száma - 1

Probléma: túl magas rendű polinom, nagyon nagy fluktuációk az ismeretlen szakaszokon. Tehát nem biztos, hogy a magasabb rendű illesztés pontosabb. Tipikus 2-d, 3-d (esetleg 4-d vagy 5-d) rendű interpoláció.

Lokális interpoláció esetén a deriváltak nem automatikusan folytonosak. Ha szükség van rá ezt külön kell biztosítani: *spline* (bütyök) illesztés.

Interpoláció és extrapláció polinommal

Az interplációnál általában nincs szükség a polinom együtthatóira. Nem is érdemes azokat kiszámolni és azok segítségével határozni meg az ismeretlen pontokat, mert sokkal érzékenyebbek a numerikus hibákra, mintha az ismeretlen pontokat direkt módon az ismertekből határozzuk meg.

N ponton kersztül egyértelműen rajzolható $N-1$ -ed rendű polinom $P(x)$. Fejezzük ki a $\{x_i, y_i = f(x_i)\}_{i=1..N}$ ismert értékkel $P(x)$ -et!

$$P(x) = \frac{(x-x_2)(x-x_3)...(x-x_N)}{(x_1-x_2)(x_1-x_3)...(x_1-x_N)}y_1 + \frac{(x-x_1)(x-x_3)...(x-x_N)}{(x_2-x_1)(x_2-x_3)...(x_2-x_N)}y_2 + \cdots \frac{(x-x_1)(x-x_2)...(x-x_{N-1})}{(x_N-x_1)(x_N-x_2)...(x_N-x_{N-1})}y_N$$

Kielégíti a $P(x_i) = y_i$ egyenlőséget, hiszen az i -ik tagban a számláló és nevező azonos, a többi tagban pedig a számláló 0. A fenti ún. *Lagrange formulá*t előállíthatjuk rekurzív módon (*Neville algoritmus*) segítségével.

Legyen $P_{i(i+1)...(i+m)}$ egy olyan m -ed rendű polinom amely átmegy az $i, (i+1), ...(i+m)$ pontokon. Ez előállítható az alábbi módon:

$$P_{i(i+1)...(i+m)} = \frac{(x-x_{i+m})P_{i(i+1)...(i+m-1)} + (x_i-x)P_{(i+1)(i+2)...(i+m)}}{x_i-x_{i+m}}$$

A két elődpolinom közül az mindkettő átmegy az $(i+1), (i+2)...(i+m-1)$ pontokon, a két szélső ponton pedig egyikük illetve másikuk. A rekurzió kiindulása azok a 0 -d rendű polinomok amelyek átmennek a az i -ik ponton, vagyis $P_i = y_i$ Definiáljuk a polinom és két elődjének eltérését:

$$C_{m,i} = P_{i(i+1)\ldots(i+m)} - P_{i(i+1)\ldots(i+m-1)}$$
$$D_{m,i} = P_{i(i+1)\ldots(i+m)} - P_{(i+1)(i+2)\ldots(i+m)}$$

Ezekre felírva a rekurzív formulát:

$$C_{m+1,i} = \frac{(x_i - x)(C_{m,i+1} - D_{m,i})}{x_i - x_{i+m+1}}$$
$$D_{m+1,i} = \frac{(x_{i+m+1} - x)(C_{m,i+1} - D_{m,i})}{x_i - x_{i+m+1}}$$

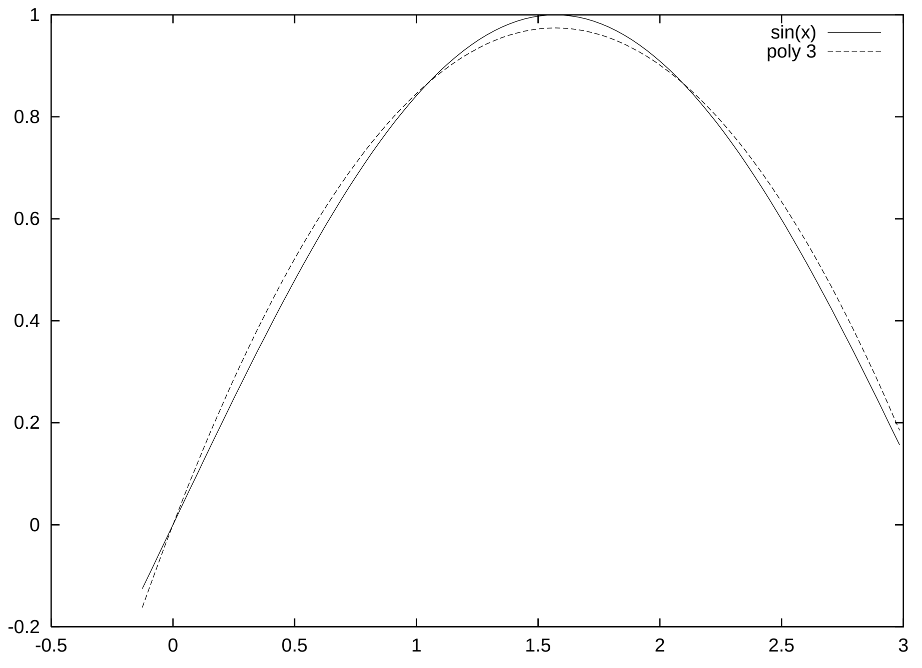
C és D korrekciót ad a polinomhoz, és eggyel magasabbrendűvé alakítja. Ha a korrekciók nem csökkennek, akkor az alapfeltevés, miszerint a pontokat polinommal érdemes interpolálni, nem áll fenn. Az utolsó lépésben alkalmazott korrekciót tekinthetjük egyfajta hibabecslésnek

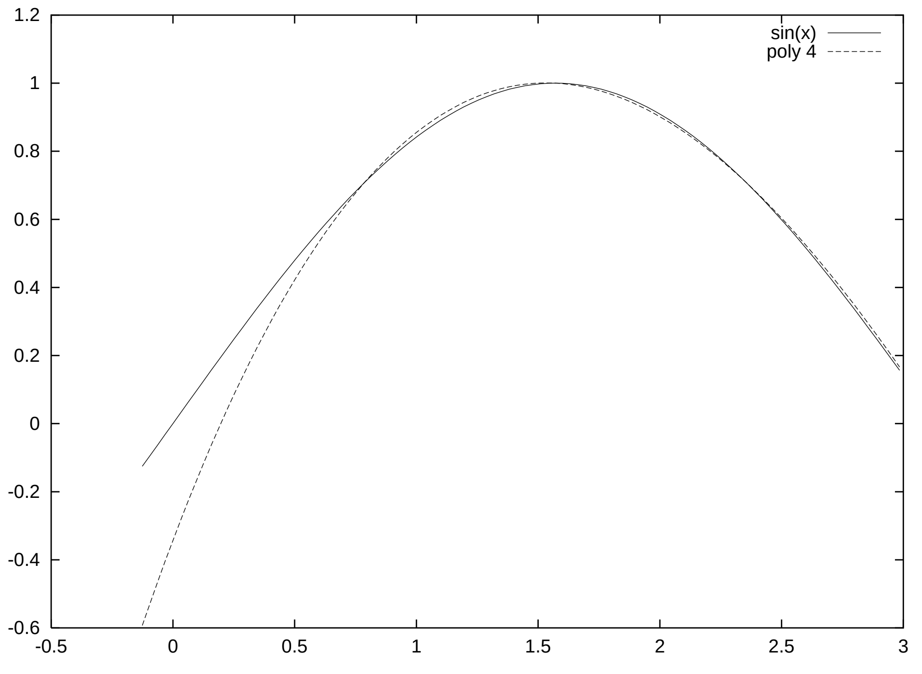
```
#include <math.h>
#include "nrutil.h"

void polint(float xa[], float ya[], int n, float x, float *y,

float *dy)
{
int i,m,ns=1;
float den,dif,dift,ho,hp,w;
float *c,*d;

dif=fabs(x-xa[1]);
c=vector(1,n);
d=vector(1,n);

/* Megkeressük az x-hez legközelebbi xa[ns]-t */
for (i=1;i<=n;i++) {
if ( (dift=fabs(x-xa[i])) < dif) {
ns=i;
dif=dift;
}
c[i]=ya[i];
d[i]=ya[i];
}
*y=ya[ns--];
for (m=1;m<n;m++) {
/* Kiszámoljuk c-t és d-t */
for (i=1;i<=n-m;i++) {
ho=xa[i]-x;
```

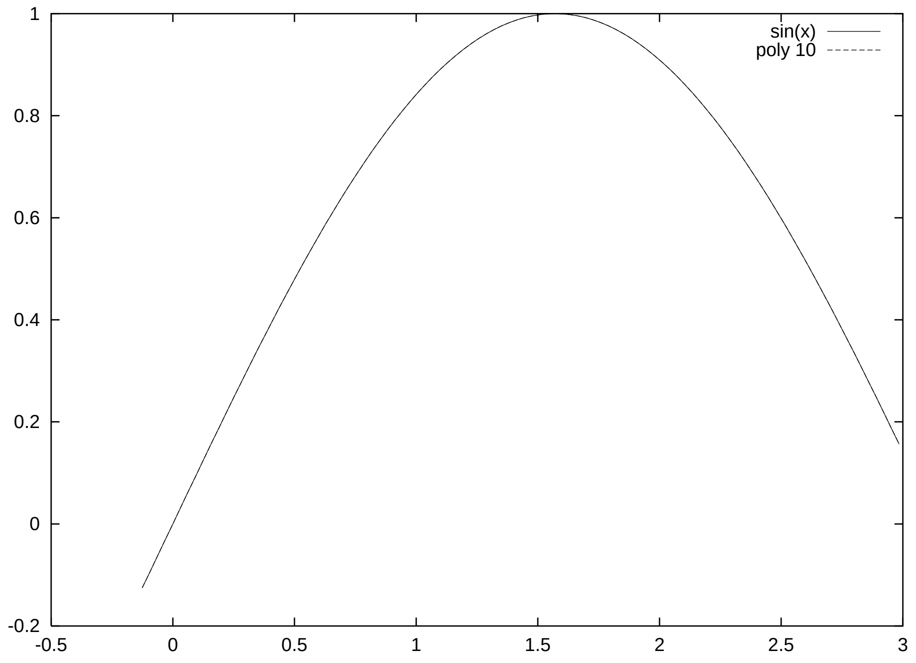
```
hp=xa[i+m]-x;
w=c[i+1]-d[i];
if ( (den=ho-hp) == 0.0)
nrerror("Error in routine polint: identical x values");
den=w/den;
d[i]=hp*den;
c[i]=ho*den;
}
```
/* Az ismeretlen pontot amennyire lehet középen tartva hozzáadjuk a korrekciót y-hoz*/

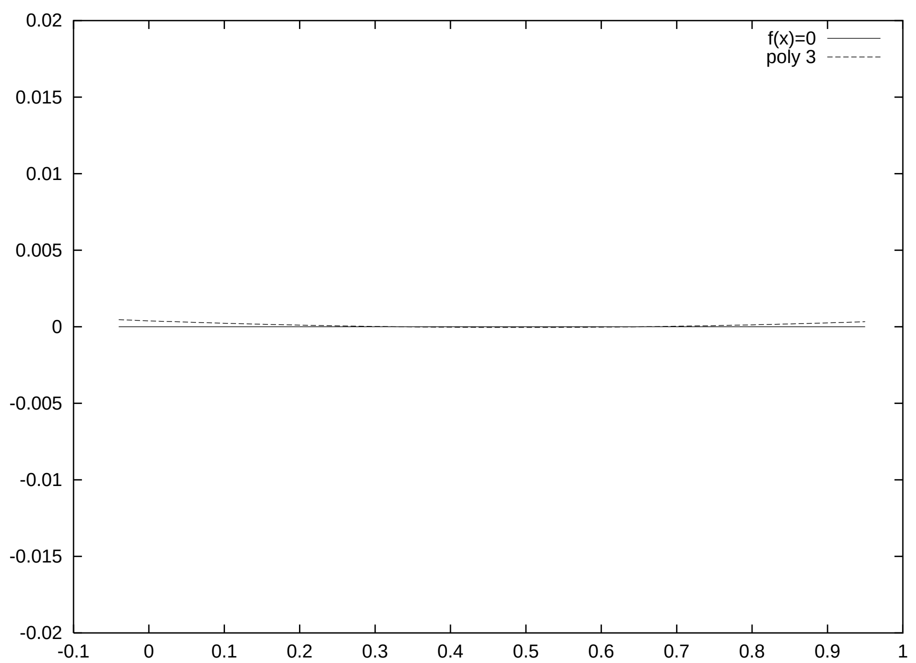
```
*y += (*dy=(2*ns < (n-m) ? c[ns+1] : d[ns--]));

}

free_vector(d,1,n);
free_vector(c,1,n);
}
```

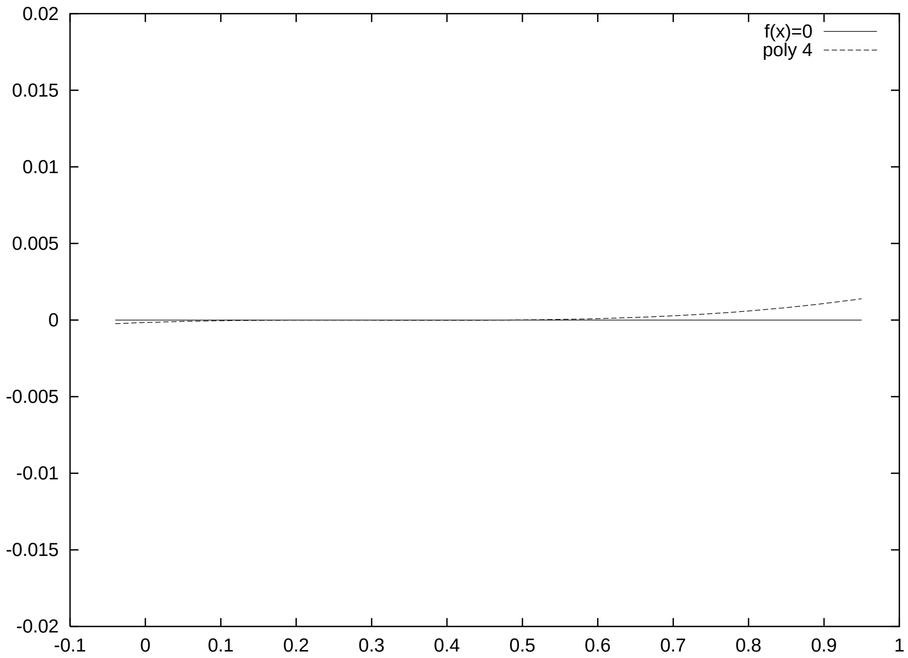
Egyes függvények jól közelíthetőek polinommal (lásd hatványsorba fejtés) mások nem. Figyelni kell! (Példák: $f(x) = sin(x), f(x) = 0 + 0.01 * rnd()$).



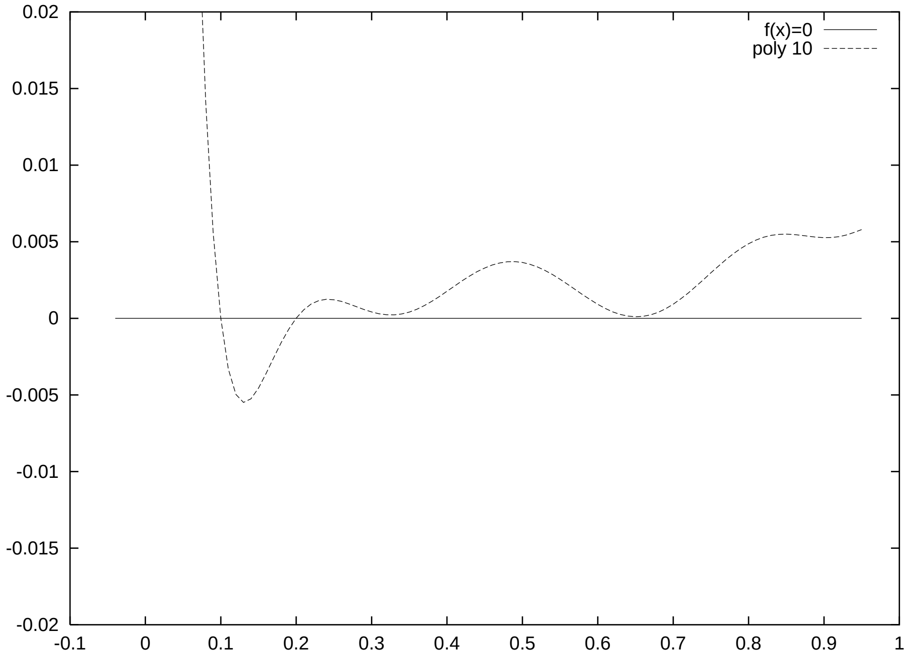



sin(x)
poly 10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.5
0
0.5
1
1.5
2
2.5
3


f(x)=0
poly 3
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



f(x)=0
poly 4



f(x)=0
poly 10

A polinom együtthatói

Általában az előző módszert célszerű használni, mert sokkal kevésbé érvényes a numerikus hibákra.

Ha minden adott pontra felírjuk, akkor az együtthatókra a következő lineáris egyenletrendszert kapjuk:

$$y_i = c_0 + c_1 x_i + c_2 x_i^2 + ... + c_N x_i^N \qquad i = 1...(N+1)$$

Az egyenletrendszer megoldható általános lineáris egyenletrendszer megoldó algoritmussal is, de van hatékonyabb speciális módszer is (lásd később: Vandermonde mátrix).

Ha az egyenletrendszer megoldása numerikus problémák miatt nem működik, használható egy lassabb módszer is. Ha az előző interpolációs módszer segítségével meghatározzuk az interpolált (extrapolált) értéket a 0 pontban, akkor pont c_0 -t kaptuk meg. Ha most minden y_i -ből levonunk c_0 -t és leosztunk a megfelelő x_i -kkel, és egy pontot elhagyunk, akkor az eredetivel megegyező problémához jutottunk, csak egyel kevesebb együtthatóval, egyel alacsonyabb rendű polinommal.

Interpoláció racionális függvénnyel

Sok függvény nem jól közelíthető polinommal, viszont polinomok hányadosával igen.

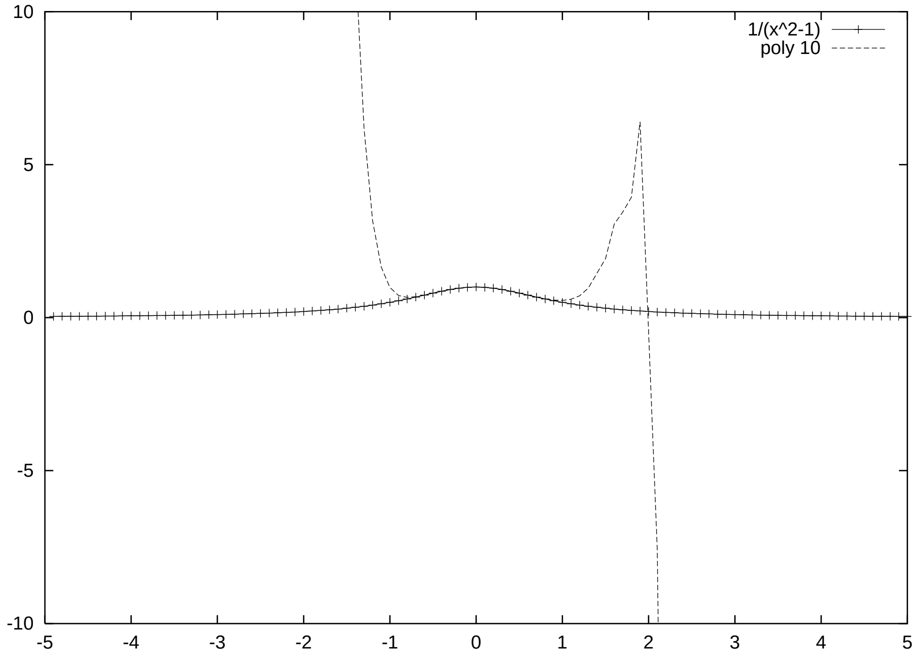
$$R_m(x) = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0+p_1x+\ldots+p_\mu x^\mu}{q_0+q_1x+\ldots+q_\nu x^\nu}$$

Itt P_μ egy μ -ed rendű Q_ν pedig egy ν -ed rendű polinom. Így összesen $m+1 = \mu+\nu+1$ szabad paraméterünk van, ha $m+1$ pontra illesztünk (hiszen ha p_0/q_o rögzített, akkor értékük szabadon választható).

Ha az interpolálandó függvénynek pólusai vannak (divergál), jobb közelítés mint a polinom. De ha nincs valós pólus, viszont a függvény analitikus kiterjesztésének vannak pólusai a komplex síkon a megadott pontokhoz közel akkor is nagyon rossz lehet a polinom közelítés. Ilyen függvények hatványsorának a konvergenciájával problémák vannak. A számláló és nevező rendje a problémától függően választható.

Pl. az alábbi függvénynek pólusa van a i -ben de a valós részének nincs. A ploinom még magas rendben sem képes jól interpolálni.

$$f(x) = \frac{1}{x^2+1}$$



Hasonló rekurzív formula felírható (HF: lásd a könyvben) mint a polinom interpolációnál.

Köbös spline interpoláció

Eddig olyan módszerekről volt szó amelyek az összes adott pontot interpolálják. Gyakran van szükség kellően sima interpolációra, de nem szükséges egy explicit polinom alak. Legegyszerűbb a pontpáronkénti lineáris interpoláció (a Lagrange formula speciális legegyszerűbb alakja) :

$$y = Ay_j + By_{j+1}$$

$$A = \frac{x_{j+1} - x}{x_{j+1} - x_j}$$

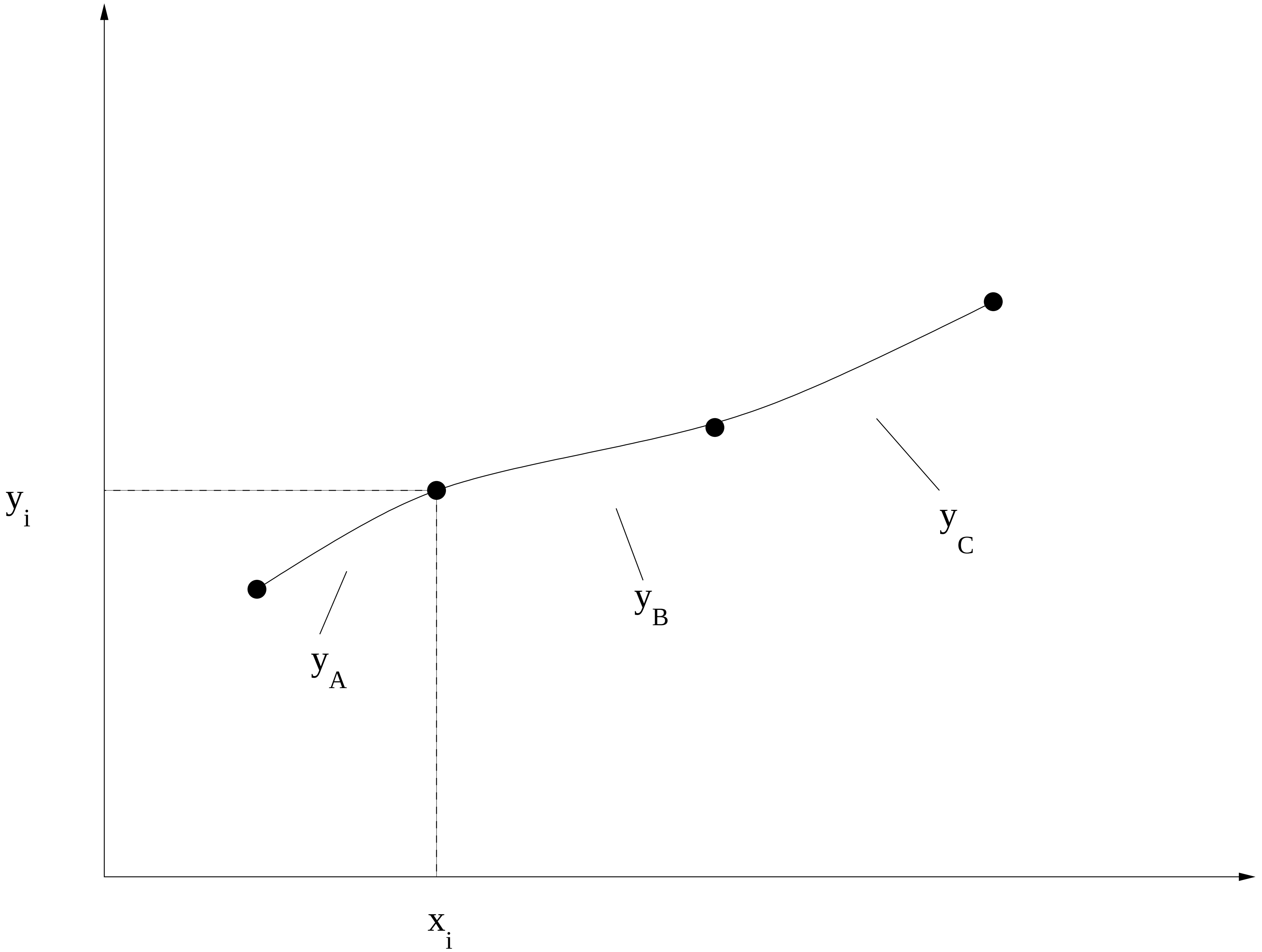
$$B = 1 - A = \frac{x - x_j}{x_{j+1} - x_j}$$

(Kijön egyszerűen a lineáris egyenletéből is:

$$y = y_j + (x - x_j)\frac{y_{j+1} - y_j}{x_{j+1} - x_j}$$

átrendezéssel) Tekinthetjük úgy is, hogy keresük az A és B lineáris polinomokat, amelyek az y_j ill. y_{j+1} lineáris együtthatókkal kombinálva adják meg a kívánt interpoláló függvényt.

Gyakran ez is elegendő azonban a megadott x_i pontokban nem léteznek az 1. és 2. deriváltak. A spline interpoláció célja, hogy sima első driváltja és folytonos második deriváltja legyen az interpolált függvénynek.



Szeretnénk, hogy

$$y_A(x_i) = y_i$$

$$y_B(x_i) = y_A(x_i)$$

$$\frac{\partial y_A}{\partial x}(x_i) = \frac{\partial y_B}{\partial x}(x_i)$$

$$\frac{\partial^2 y_A}{\partial x^2}(x_i) = \frac{\partial^2 y_B}{\partial x^2}(x_i)$$

Ez 4 egyenlet, negyedrendű interpoláció lehetséges, vagyis pl. egy harmadrendű polinom. Ebből nem túl szemléletes az interpolációs polinom levezetése ezért inkább tegyük fel, hogy ismerjük minden x_i pontban a második deriváltakat ($y_i^{''}$) is. Akkor ha hramdrendű polinomot választunk az interpolációhoz, akkor egy (x_i, x_{i+1}) pontpárra 4 lineáris egyenletet kapunk, aminek egyértelmű megoldása lehet.

Keressük a következő formában fel az interpolációs függvényt:

$$y = Ay_j + By_{j+1} + Cy_j^{''} + Dy_{j+1}^{''}$$

ahol A, B, C, D max. harmadrendű polinomok. A és B legyen a fenti érték és

$$C = \tfrac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2$$

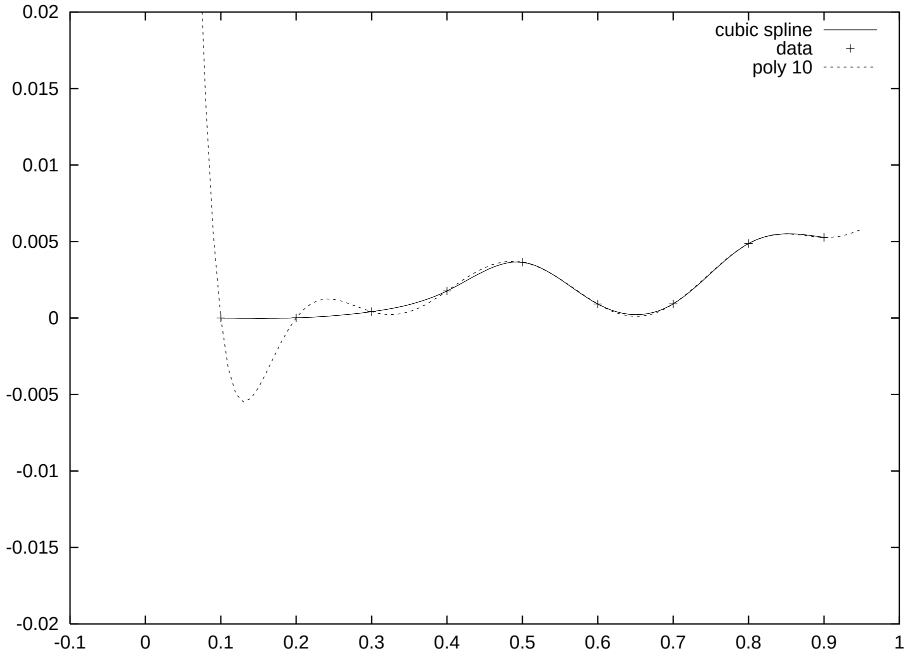
$$D = \tfrac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2$$

adódik. Leellenőrizhető hogy ennek második deriváltjai valóban $y_j^{''}, y_{j+1}^{''}$ és a függvényérték is stimmel. A második deriváltakat nem ismerjük eleve, viszont ha megköveteljük, hogy a két szélső pont kivételével minden pontban a jobb és baloldali első deriváltak megegyezzenek kapunk $N-2$, $y_i^{''}$ -kben lineáris egyenletet.

$$\frac{x_j-x_{j-1}}{6}y_{j-1}^{''}+\frac{x_{j+1}-x_{j-1}}{3}y_j^{''}+\frac{x_{j+1}-x_j}{6}y_{j+1}^{''}=\frac{y_{j+1}-y_j}{x_{j+1}-x_j}-\frac{y_j-y_{j-1}}{x_j-x_{j-1}}$$

Ha még két szabad paramétert rögzítünk, akkor a kapott egyenletrendszer megoldásával egyértelmű a második deriváltak értéke. Rögzíthetjük pl. 0 -nak $y_1^{''}$ -t és $y_N^{''}$ -t, vagy megadhatjuk pl. a határokon az első derivált értékét.

Ez a spline illesztés tehát egy $N-2$ ismeretlenes lineáris egyenlet megoldásával jár. Viszont szerencsére egyszerű ún. *tridiagonális* egyenletrendszer, vagyis együegy egyenletben egy ismeretlen mellett csak a két szomszédos indexű ismeretlen van. Az ilyen lineáris egyenletrendszerekre elég hatékony megoldás ismert (lásd később).

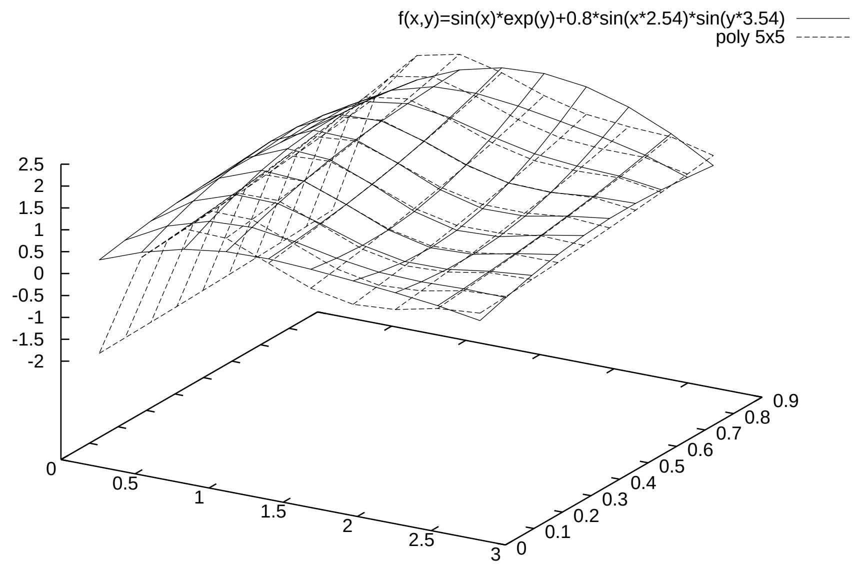


Interpoláció több dimenzióban

Itt csak a legegyszerűbb 2 dimenziós esetet tárgyaljuk négyzetrácson ismert pontokkal. A több dimenziós eset teljesen analóg, a random eloszlású ismert pontokra való interpoláció bonyolultabb (Delanuey háromszögelés, Voronoi cellák).

Indexeljük a rácsot i, j -vel. Ismert $((x, y)_{ij}, z_{ij})$. Szeretnénk meghatározni z -t egy tetszőleges (x, y) pontban.

Polinom illesztés: A rács egyik pl. x irányában illesztünk minden ismert y_j értéknél $z_j = P_{y_j}(x)$ polinomokat. Most az így kapott polinomoknak meghatározzuk az értékét a kívánt pont x értékénél. Ezeken a pontokon interpolálunk immár y irányban egy újabb $z = P_x(y)$ polinomot, mely y helyen felvett értéke adja a kívánt z -t.



Ugyanezt megtehetjük polinom helyett köbös spline görbékkel. (bicubic spline)

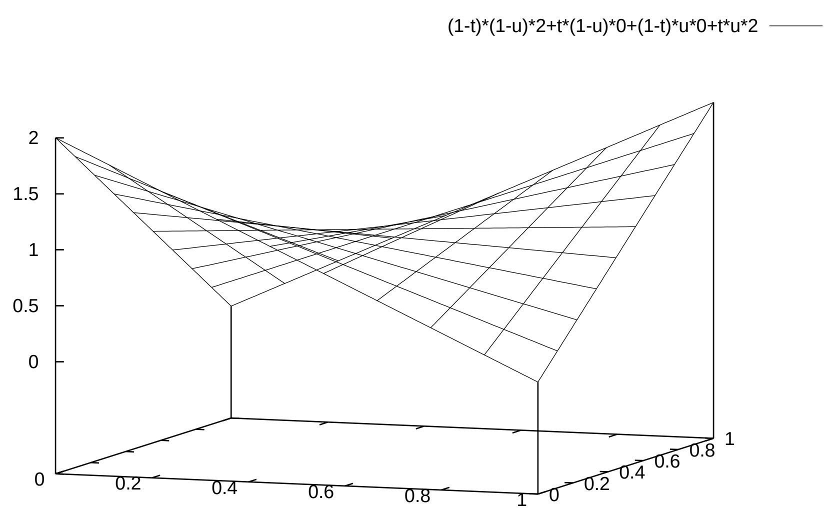
Bilineáris interpoláció: Hasonlóan az 1 dimenziós esethez, határozzuk meg, hogy a meghatározandó pont x illetve y koordinátája hogyan viszonyul az őt bekeretező 4 pont koordinátáihoz:

$$t = (x - x_i)/(x_{i+1} - x_i)$$

$$u = (y - y_j)/(y_{j+1} - y_j)$$

A z_{ij} értékek azonos arányú lineáris kombinációja adja a kívánt értéket:

$$z = (1-t)(1-u)z_{ij} + t(1-u)z_{(i+1)j} + (1-t)uz_{i(j+1)} + tuz_{(i+1)(j+1)}$$



Ez is folytonos felületet ad, viszont nem sima. Hasonlóan az 1 dimenziós esethez, itt is megkövetelhetjük még további feltételként a deriváltak folytonosságát is, viszont nem kapunk egyértelmű egyenleteket a második deriváltakra, mint az előbb. Szükséges $\partial z/\partial x$, $\partial z/\partial y$ és $\partial^2 z/\partial x \partial y$ deriváltak explicit megadása. A deriváltak értékétől függetlenül a felület sima lesz, viszont nem garantált, hogy jól közelíti a kívánt függvényt.