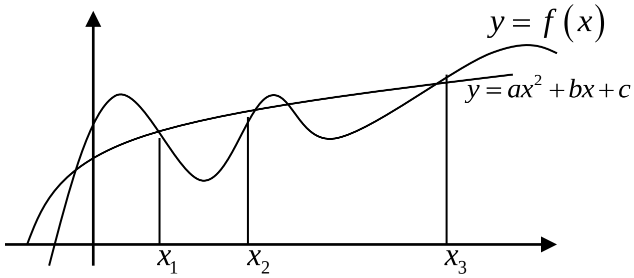
אינטרפולציה



פעולה המאפשרת לקרב את התוצאה בעזרת פולינומים. נבחר אוסף של נקודות $x_0, x_1, ..., x_n$ $y_0, y_1, ..., y_n$ כאשר $y_i = f(x_i)$. נעביר קווים. הנקודות הללו נקראות נקודות אינטרפולציות.

את הנקודות שקיבלנו נסדר בטבלה:

X	x_0	x_1	$\dots$	x_n
Y	y_0	y_1	$\dots$	y_n

כאשר יש n נקודות נתונות, יש פולינום אחד ויחיד מסדר n שעובר דרכן.

ואת התוצאה נסדר במערכת משוואות:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$ (מטריצה מסדר $(n+1)$)

נכליל דטרמיננטות של מטריצות ונדרמונדה: $\det(W) = (-1)^s \prod_{i \neq j} (x_i - x_j)$

קיימת שגיאת תנאי גדולה כאשר x_i ו- x_j קרובים מאוד זה לזה.

נוכל לפתור את המטריצה בעזרת פולינום לגראנז' הפותר את הבעיה האינטרפולצית.

$$L_n(x) = \sum_{i=0}^{n} \omega_i(x) y_i \qquad \left[\omega_i = \prod_{i \neq j} \frac{(x - x_j)}{(x_i - x_j)} \right]$$

באופן כללי, $\omega_i(x_k) = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$. מספר הפעולות לאלגוריתם זה $\sim 2n^2$

השאלה היא מתי אפשר להשתמש בפולינום במקום פונקציה, ועד כמה החישובים מדויקים. (הדיוק סביב נקודות האינטרפולציה הן טובות מאוד, אולם הנקודות האחרות פחות מדויקות)

הפולינום של ניוטון פותר את אותה הבעיה. $L_n(x) = N_n(x)$ (ישנו רק פתרון אחד).

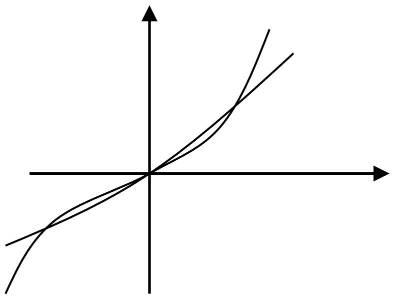
$$f[x_i] = f(x_i) = y_i$$

$$f[x_1, x_2] = \frac{f[x_1] - f[x_2]}{x_1 - x_2}$$

$$f[x_1, ..., x_n] = \frac{f[x_0, ..., x_{n-1}] - f[x_1, ..., x_n]}{x_0 - x_n}$$

ואז: $L_n(x) \equiv N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + ... f[x_0, ..., x_k](x - x_0) \cdot \ldots \cdot (x - x_{k-1})$.

<u>תרגיל:</u> נתונה הפונקציה $f(x) = x^3$. נקרב אותה ע"י פרבולה.



$x_0 = -1$

$x_1 = 0$

$x_2 = 1$

נפתור זאת לפי האלגוריתם של ניוטון.

$N_2(x) = 1 + 1(x+1) + 0(x+1)\cdot x = x$ זהו הפולינום מסדר 1

למרות שהוא יחיד בין פולינומים מסדר 2.

נפתור גם ע"י פולינום לגראנז':

$$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} y_0 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} y_2 = \frac{x(x-1)}{-1(-2)}(-1) + \frac{(x+1)x}{1\cdot 2} =$$

$$= -\frac{x^2-x}{2} + \frac{x^2+x}{2} = x$$

האלגוריתם של ניוטון דורש $n + (n-1) + (n-2) + \ldots + 1 \sim \frac{n^2}{2}$ פעולות.

אם נקבל מידע בצורה רקורסיבית (כלומר, בכל עמודה בטבלת הערכים יתווסף עוד נתון אחד) אז יש להוסיף רק n צעדים לאלגוריתם ניוטון, ומכאן שעדיף להשתמש בו.

בשני הפולינומים (הזהים) ישנה אותה השגיאה! אפשר למדוד את השגיאה לפי

$\left|f(x) - P_n(x)\right| \le \left|\frac{f^{(n+1)}(\xi)}{(n+1)!}\right| \left|(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_n)\right|$ ($\xi =$ 'קסיי', נקודה באמצע תחום האינטרפולציה).

$$L = \max_{x_0 \le x \le x_n} \left\{(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_n)\right\} = \max\left\{x^3 - x\right\}$$

$$\Rightarrow (x^3 - x)' = 0 \Rightarrow 3x^2 - 1 = 0 \Rightarrow x = \pm\frac{1}{\sqrt{3}}$$

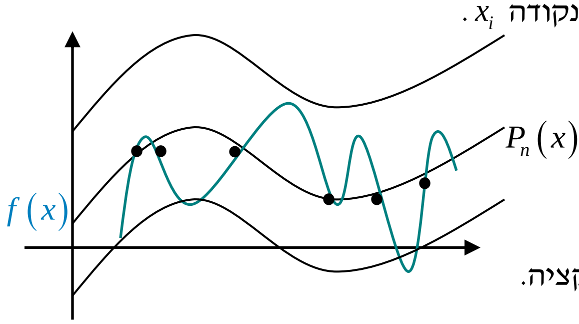
ב-*Matlab*, אם נרצה להעביר פולינום בין נקודות X,Y כלשהן, נכתוב את הפקודות הבאות:

$[P,S] = polyfit(X,Y,n)$

כאשר P ייצג $[a_0, a_1, ..., a_n]$, הנקודות X,Y ייוצגו ע"י וקטורים, ו-n הוא המעלה של הפולינום.

הפקודה $polyval(p,X)$ תיתן לנו את ערך הפולינום p בנקודה X (שתיוצג ע"י וקטור).

הפקודה $interp1(X,Y,x_i)$ תיתן את ערך הפונקציה בנקודה x_i.



בעיה: כיצד נוכל לבחור נקודות אינטרפולציה

$x_0, \ldots, x_n$ כך ש- $\max_{x_0 \le x \le x_n} \left|w_{n+1}(x)\right| \to \min$?

שקול לשאול: כיצד נוכל למצוא פולינום $P_n(x)$

כך ש- $\left\|x^{n+1} - P_n(x)\right\|_{c[a,b]} \to \min$.

כלומר, מהו הפולינום הטוב ביותר והקרוב ביותר לפונקציה.

<u>משפט צ'ביצ'ב:</u> $P_n(x)$ נותן את הקירוב הטוב ביותר לפונקציה רציפה בקטע $[a,b]$ $\Leftrightarrow$

יש $n+2$ נקודות $x_1 \le x_2 \le \ldots \le x_{n+2}$ המקיימות את התנאי צ׳ביצ׳ב:

1. $f(x_i) - P_n(x_i) = (-1)^i \varepsilon$ לכול $i = 1,2,\ldots,n+2$
2. $\left\| f(x) - P_n(x) \right\|_{C[a,b]} = \left| f(x_i) - P_n(x_i) \right| = \varepsilon$

לדוגמא: לפי תנאי צ׳ביצ׳ב, מבין כל הפונקציות הליניאריות, הקירוב הכי טוב ל- x^3 הוא $\frac{3}{4}x$. כלומר, $\left[x^3 - \frac{4}{3}x \right]_{c[-1,1]}$ יהיה מינימאלי.

$$\left(x^3 - \frac{3}{4}x \right)' = 0 \Rightarrow 3x^2 - \frac{3}{4} = 0$$

$$x = \pm \frac{1}{2}$$

ניקח $x_1 = -1, x_2 = \frac{1}{2}, x_3 = \frac{1}{2}$ (כי במקרה זה $n+2=3$).

x	$x^3 - \frac{3}{4}x$
$x_1 = -1$	$-1/4$
$x_2 = -1/2$	$1/4$
$x_3 = 1/2$	$-1/4$

$\Leftarrow$ בחרנו 3 נקודות אקסטרימליות (ההפרש לא גדול מ- $1/4$).

$$\left\| x^3 - \frac{3}{4}x \right\| = dist\left(x^3, P_1 \right) = \frac{1}{4}$$

ואי אפשר לשפר את זה. אגב, גם מבין כל הפרבולות, $P_2 = \frac{3}{4}x$ תהיה הקירוב הטוב ביותר.

אם נרצה למצוא את הקירוב הטוב ביותר ל- x^{n+1} ע״י פולינום $P_n(x)$ בקטע $C[-1,1]$ נפעל כך:

$$P_n(x) = \frac{1}{2^{n-1}} \cdot T_n(x)$$

$$T_n(x) = \cos\left(n \cdot \arccos x \right)$$

נבדוק שזהו אכן פולינום:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$\cos\left((n+1)\alpha \right) + \cos\left((n-1)\alpha \right) = 2\cos\left(n\alpha \right) \cdot \underset{\alpha = \arccos x}{\cos\alpha}$$

$$T_{n+1}(x) + T_{n-1} = 2T_n \cdot x$$

$\Leftarrow$ קיבלנו נוסחה רקורסיבית.

$$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$$

ולכן תמיד נישאר בפולינומים. (פולינום זה מקיים את התנאי של צ׳ביצ׳ב)

$$\left\|x^{n+1}-P_n(x)\right\| \to \min \Leftrightarrow P_n(x)=\frac{1}{2^n}\cdot T_{n+1}(x)$$

$$\Rightarrow (x-x_0)\ldots(x-x_n)=w(x)=\frac{1}{2^n}\cdot T_{n+1}(x)$$

ולכן את האינטרפולציה נבחר כשורשים של פולינום צ׳ביצ׳ב מאותה מעלה!

איך נקרב פולינום כללי?

אנו צריכים לבנות אלגוריתם שיקיים: $\left\|f(x)-ax^2-bx-c\right\|_{C[-1,1]} \to \min$.

נגזור ונמצא נקודת מינימום: $f'(x)-2ax-b=0$ ונקבל פתרונות $-1,x_1,x_2,\ldots,1$.

נציב:

$$\begin{cases} f(x_1)-ax_1^2-bx_1-c=\varepsilon \\ f(x_2)-ax_2^2-bx_2-c=-\varepsilon \\ f(x_3)-ax_3^2-bx_3-c=\varepsilon \\ f(x_4)-ax_4^2-bx_4-c=\varepsilon \end{cases}$$

נפתור ונקבל את a,b,c,ε.

במקרה זה נקבל תוצאה שאי אפשר לשפר אותה! בקירוב ע״י אינטרפולציה רגילה, נקבל תוצאה שכן ניתן לשפר.

$$\varepsilon=\left\|f(x)-P_n^*(x)\right\|_{C[a,b]} \to \min$$

$$\varepsilon=dist\left(f(x),P_n,C[a,b]\right)$$

$$\left\|f(x)-\underbrace{L_n(x,x_0,\ldots,x_n)}_{Laugrange}\right\| \le \left(1+\underset{\ge 0}{\mu}\right)\cdot\varepsilon$$

אם μ, מספר לבג׳ (*Lebesque const*) גדול אז לא כדאי לעשות אינטרפולציה. אם μ קטן מאוד ונבצע את פולינום צ׳ביצ׳ב אז לא נרוויח הרבה.

הפונקציה של לבג׳:

$$Le(x)=\sum_{i=0}^{n}\left|w_i(x)\right|$$

$$\max_{[x_0\le x\le x_n]} Le(x)=\mu$$

<u>הערה:</u> פולינום צ׳ביצ׳ב עובד רק עבור $C[-1,1]$. אם יש קטע רנדומאלי $[a,b]$ אז נבנה העתקה ליניארית: $x=\frac{a-by}{2}+\frac{a+b}{2}$, ואז: $T_n=\left(\frac{a-b}{2}x+\frac{a+b}{2}\right)$. תהליך זה נקרא *dilatation* ו*translation* (הרחבה והעתקת מקום).

דוגמא עבור $x^3,C[0,1],P_1(x)$

$$\varphi(x)=x^3-3ax-b$$

$$\varphi'(x)=3x^2-3a \Rightarrow x_{1,2}=\pm\sqrt{a}$$

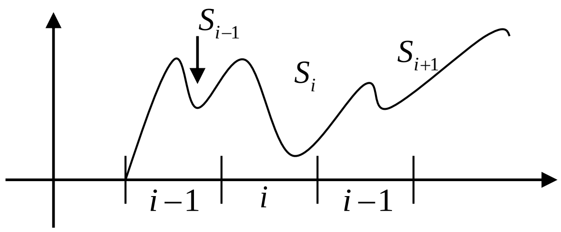
$-\sqrt{a}$ לא רלוונטי עבורנו. ולכן נקבל את התוצאה: $x=\sqrt{a}$. ניקח את הנקודות:

$a_1=0, x_2=\sqrt{a}, x_3=1$, ונבנה את המשוואות:

$$\begin{aligned}
&(1) \quad -b=\varepsilon \\
&(2) \quad \sqrt{a}\cdot a-3a\cdot\sqrt{a}-b=-\varepsilon \Rightarrow \boxed{a=\tfrac{1}{3}, b=\tfrac{-1}{\sqrt{3^3}}, \varepsilon=\tfrac{1}{\sqrt{3^3}}} \\
&(3) \quad 1-3a-b=\varepsilon
\end{aligned}$$

$dist\left(x^3,P_1\right)_{C[0,1]}=\frac{1}{\sqrt{3^3}}$, והפולינום הוא $y=\frac{1}{3}x-\frac{1}{\sqrt{3^3}}$

<u>Cubic Spline Interpolation</u>



$$S_{i-1}\left(x_i\right)=S_i\left(x_i\right)$$
$$S_{i-1}'\left(x_i\right)=S_i'\left(x_i\right)$$
$$S_{i-1}''\left(x_i\right)=S_i''\left(x_i\right)$$

S_i הם מגדרים בקטע $[x_{i-1}, x_i]$ להתאים וקיומיים כפולינומים ממעלה 3.

$$S_i\left(x\right)=a_ix_i^3+b_ix_i^2+c_ix_i+d_i=a_{i-1}x_i^3+b_{i-1}x_i^2+c_{i-1}x_i+d_{i-1}=y_i$$

וכנ״ל לגבי הנגזרות.

נקבל $4n$ משוואות, אבל לא נוריד את הקצוות. נקבל שיש $4n-4$ משוואות, ובנוסף 2 נוספות:

$$S\left(x_0\right)=y_0$$
$$S\left(x_n\right)=y_n$$

כלומר בסה״כ $4n-2$ משוואות עם $4n$ נעלמים. כדי לקבל פתרון אחד ויחיד, אנו צריכים להוסיף עוד 2 משוואות.

הפקודות ב-*Matlab* לבניית ספליינים:

`interp1`	1-D data interpolation (table lookup)
`griddedInterpolant`	Gridded data interpolation
`pchip`	Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)
`spline`	Cubic spline data interpolation
`ppval`	Evaluate piecewise polynomial
`mkpp`	Make piecewise polynomial
`unmkpp`	Piecewise polynomial details
`padecoef`	Padé approximation of time delays
`interpft`	1-D interpolation using FFT method