אינטרפולציה פולינומיאלית

$$p(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)}y_1$$

$$+\frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)}y_2$$

$$+ \\ \vdots$$

$$+\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}y_n$$

פתרון כללי:
נוסחת
Lagrange

אלגוריתם של Neville

$x_1 : \quad y_1 = p_1$

p_{12}

$x_2 : \quad y_2 = p_2$

p_{123}

p_{23}

p_{1234}

$x_3 : \quad y_3 = p_3$

p_{234}

p_{34}

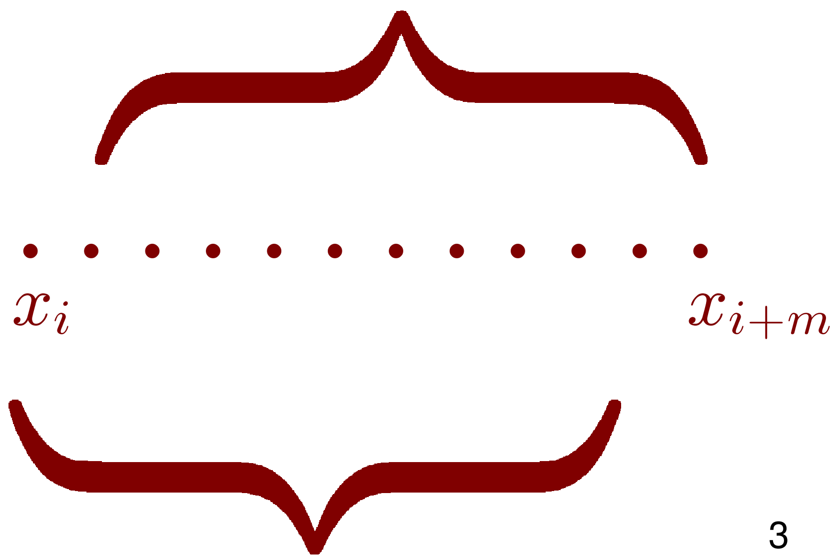
$x_4 : \quad y_4 = p_4$

פתרון כללי
יותר יעיל

פתרון רקורסיבי

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

$$p_{12} = \frac{(x-x_2)p_1 + (x_1 - x)p_2}{x_1 - x_2}$$



יותר מדויק- הבדלים בין הורים לבת

$$C_{m,i} = p_{i(i+1)\dots(i+m)} - p_{i(i+1)\dots(i+m-1)}$$

$$D_{m,i} = p_{i(i+1)\dots(i+m)} - p_{(i+1)\dots(i+m)}$$

$$\Rightarrow$$

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

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

אינטרפולציה חלקה: Cubic Spline

http://www.math.ucla.edu/~baker/java/hoefer/Spline.htm

השיטה הפרקטית: יחסית פשוטה, אבל בד"כ יעילה ומדויקת.

מתחילים מאינטרפולציה ליניארית בחלקים:

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

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

אינטרפולציה חלקה: Cubic Spline

משפרים בעזרת פולינום מסדר 3 בכל חלק. קודם,
נעמיד פנים שאנחנו יודעים את הנגזרות השניות:

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

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

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

$$\frac{d^2y}{dx^2} = Ay''_j + By''_{j+1}$$

אינטרפולציה חלקה: Cubic Spline

מוצאים גם את הנגזרת הראשונה:

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

$$\frac{d^2y}{dx^2} = Ay''_j + By''_{j+1}$$

$$\frac{dy}{dx} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{3A^2-1}{6}(x_{j+1} - x_j)y''_j + \frac{3B^2-1}{6}(x_{j+1} - x_j)y''_{j+1}$$

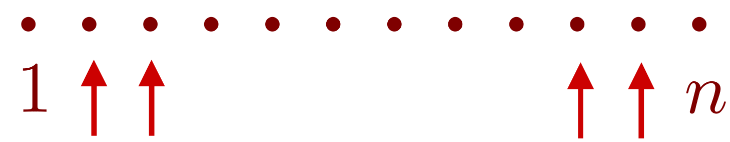
נגזרת ראשונה רציפה:

עכשיו קובעים את הנגזרות השניות, כך שהנגזרות 1 ו-2 יהיו רציפות. התנאי לרציפות הנגזרת ה-1 בנקודה x_j :

$$\frac{x_j - x_{j-1}}{6} y''_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} y''_j + \frac{x_{j+1} - x_j}{6} y''_{j+1}$$

$$= \frac{y_{j+1} - y_j}{x_{j+1} - x_{j-1}} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$

n נעלמים, n-2 תנאים (נקודות שבהן הנגזרת רציפה):



<u>**לכן, עוד שני תנאי שפה:**</u>

1. $y_1'' = 0$, **או** y_1' **נתון.**
2. $y_n'' = 0$, **או** y_n' **נתון.**

spline טבעי

אינטרפולציה: פונקציות ספרייה

void polint(float xa[], float ya[], int n, float x, float *y, float *dy)
Given arrays xa[1..n] and ya[1..n], and given a value x, this routine returns a value y, and an error estimate dy. If $P(x)$ is the polynomial of degree $N - 1$ such that $P(xa_i) = ya_i$, $i = 1, \ldots, n$, then the returned value $y = P(x)$.

void spline(float x[], float y[], int n, float yp1, float ypn, float y2[])
Given arrays x[1..n] and y[1..n] containing a tabulated function, i.e., $y_i = f(x_i)$, with $x_1 < x_2 < \ldots < x_N$, and given values yp1 and ypn for the first derivative of the interpolating function at points 1 and n, respectively, this routine returns an array y2[1..n] that contains the second derivatives of the interpolating function at the tabulated points x_i. If yp1 and/or ypn are equal to 1×10^{30} or larger, the routine is signaled to set the corresponding boundary condition for a natural spline, with zero second derivative on that boundary.

void splint(float xa[], float ya[], float y2a[], int n, float x, float *y)
Given the arrays xa[1..n] and ya[1..n], which tabulate a function (with the xa_i's in order), and given the array y2a[1..n], which is the output from spline above, and given a value of x, this routine returns a cubic-spline interpolated value y.

איתור האינטרוול: שיטת החצייה

שימוש במעבד הראשוני

```
#define  name  replacement-text
#define SIZE 100
double a[SIZE],b[SIZE];
#define Pi 3.14159265358979323 85
#define Area(r) Pi*r*r
```

שגיאה!!!

```
x=Area(s+1);
x=Area(s++);
```

סדר הפעולות עדיין לא מוגדר:

```
float sqrarg;
#define SQR(a) (sqrarg=(a),sqrarg*sqrarg)
SQR(1.)*SQR(2.) => 1.
```

Numerical Recipes :1 פרק

```
#define SQR(a) \
   ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
```

SQR(a)	**Square a float value.**
DSQR(a)	**Square a double value.**
FMAX(a,b)	**Maximum of two float values.**
FMIN(a,b)	**Minimum of two float values.**
DMAX(a,b)	**Maximum of two double values.**
DMIN(a,b)	**Minimum of two double values.**
IMAX(a,b)	**Maximum of two int values.**
IMIN(a,b)	**Minimum of two int values.**
LMAX(a,b)	**Maximum of two long values.**
LMIN(a,b)	**Minimum of two long values.**

Numerical Recipes :1 פרק

float *vector(long nl, long nh)
Allocates a float vector with range [nl..nh].
int *ivector(long nl, long nh)
Allocates an int vector with range [nl..nh].
unsigned long *lvector(long nl, long nh)
Allocates an unsigned long vector with range [nl..nh].
double *dvector(long nl, long nh)
Allocates a double vector with range [nl..nh].

float **matrix(long nrl, long nrh, long ncl, long nch)
Allocates a float matrix with range [nrl..nrh][ncl..nch].
double **dmatrix(long nrl, long nrh, long ncl, long nch)
Allocates a double matrix with range [nrl..nrh][ncl..nch].
int **imatrix(long nrl, long nrh, long ncl, long nch)
Allocates an int matrix with range [nrl..nrh][ncl..nch].

הערות נוספות:

עיגול:

הטלה cast:

(type-name) expression

```
int round(float x) {
  return (int) ((x < 0) ? (x-.5) : (x+.5));
}
```

<u>Numerical Recipes:</u>

float (ספרות 7)

if (w[j]) {   }

<u>אני משתמש:</u>

double (ספרות16)

if (fabs(w[j]) > 1.e-8) {   }

שגיאות עיגול וקיצוץ

ניסיון ראשון:
```
a = (x != 0) ? (exp(x)-1)/x : 1;
```

עדיף:
```
a = (fabs(x) > 1.e-5) ? (exp(x)-1)/x : 1;
```

עיגול: $\sim 10^{-7}, \quad +/-$

קיצוץ: $x/2$

exp(1.e-5)=1.0000101

a=1.01

שגיאת עיגול של 1.e-7 ב-x
הופכת ל- 1% ב-a !

$$\boxed{\texttt{a = (fabs(x) > 1.e-5) ? (exp(x)-1)/x : 1+x/2;}}$$

קיצוץ: $x^2/6$