Programování pro fyziky
Interpolace a extrapolace I/III

Jan Schee
ÚF FPF SU Opava, 2012

Předpokládejmě, že známe funkční hodnoty $f(x)$ v bodech

$$x_1 < x_2 < \cdots < x_N$$

Chceme-li určit hodnotu $f(x)$ ve kterémkoliv bodě z intervalu

$$x \in [x_1, x_N]$$

pak musíme určit **interpolaci** funkce $f(x)$ na daném intervalu. Všemi body $x_1, x_2, \ldots, x_N$ proložíme interpolační funkci tak aby co nejlépe (s co nejmenší chybou) aproximovala funkci $f(x)$.

Chleme-li určit hodnotu funkce $f(x)$ v bodě

$$x \notin [x_1, x_N]$$

pak funkci v tomto bodě **extrapolujeme**.

Interpolace vychází z následujícího schématu:

- Dvěmi body vede jednoznačně určená přímka
- Tři body jednoznačně určí parabolu
- ...

V oblasti mezi zadanými body je f(x) aproximována známými funkcemi:

- polynomy
- racionální funkce
- trigonometrické funkce

Interpolační polynom řádu N-1 procházející N body $y_1=f(x_1)$, $y_2=f(x_2),\ldots,y_N=f(x_N)$ je dán Lagrangeovou formulí

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

Tuto formuli numericky efektivně implementuje tzv.**Nevillův algoritmus**

Schéma Nevillova algoritmu pro N=4

$$
\begin{array}{lllll}
x_1: & y_1 = P_1 & & & \\
 & & P_{12} & & \\
x_2: & y_2 = P_2 & & P_{123} & \\
 & & P_{23} & & P_{1234} \\
x_3: & y_3 = P_3 & & P_{234} & \\
 & & P_{34} & & \\
x_4: & y_4 = P_4 & & &
\end{array}
$$

Rekurzivní formule Nevillova algoritmu

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

která ,např., pro m=1 přejde na tvar

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

pro i=1 pak obdržíme výraz

$$P_{12} = \frac{(x - x_2) P_1 + (x_1 - x) P_2}{x_1 - x_2}$$

```
void polint(double xa[], double ya[], int n, double x, double *y, double *dy)
{
  double *P;
  double *PP;
  int i,j,m,cnt;

  P=(double *)malloc((n+1)*sizeof(double));
  PP=(double *)malloc((n+1)*sizeof(double));

  for(i=1;i<=n;i++) P[i]=ya[i];/*inicializace prvniho sloupce*/

  cnt=n-1;
  for(i=1;i<=n-1;i++)  /* smycka pres zbyle sloupce*/
  {
    m=n-cnt;

    for(j=1;j<=cnt;j++)
    {
      PP[j]=((x-xa[j+m])*P[j]+(xa[j]-x)*P[j+1])/(xa[j]-xa[j+m]);
    }

    for(j=1;j<=cnt;j++) P[j]=PP[j];

    /*kazdy sloupec ma o jeden prvek mensi pocet nez predchozi sloupec*/
    cnt=cnt-1;
  }

  /*v poslednim sloupci je jeden prvek a v nem je vysledna aproximace*/
  *y=PP[1];

  free(P);
  free(PP);
}
```

Známe funkční hodnoty y_i ve vybraných bodech x_i, kterých je N.

Určete hodnotu y v libovolném bodě x užitím lineární interpolace. Tzn. mezi každými sousedními body aproximujete funkci y přímkou

$$y = \frac{(y_i - y_{i+1})x + (y_{i+1}x_i - y_i x_{i+1})}{x_i - x_{i+1}}$$