UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

# BAKALÁŘSKÁ PRÁCE

Interpolace v matematickém softwaru R

**Katedra matematické analýzy a aplikací matematiky**
Vedoucí bakalářské práce: **RNDr. Jitka Machalová, Ph.D.**
Vypracovala: **Ingrid Straussová**
Studijní program: B1103 Aplikovaná matematika
Studijní obor: Matematika–ekonomie se zaměřením na bankovnictví/pojišťovnictví
Forma studia: prezenční
Rok odevzdání: 2017

# BIBLIOGRAFICKÁ IDENTIFIKACE

**Autor:** Ingrid Straussová

**Název práce:** Interpolace v matematickém softwaru R

**Typ práce:** Bakalářská práce

**Pracoviště:** Katedra matematické analýzy a aplikací matematiky

**Vedoucí práce:** RNDr. Jitka Machalová, Ph.D.

**Rok obhajoby práce:** 2017

**Abstrakt:** Tato bakalářská práce je zaměřená na polynomiální interpolaci. V jednotlivých kapitolách jsou představeny metody pro nalezení interpolačního polynomu. Každá metoda je doplněna vlastním, podrobně vysvětleným příkladem. Práce je doplněna dávkovými soubory vytvořenými v matematickém softwaru R. Kódy v těchto dávkových souborech jsou naprogramovány tak, aby našly koeficienty interpolačního polynomu, a pro tento interpolační polynom vykreslily graf.

**Klíčová slova:** interpolace, polynomiální interpolace, Lagrangeova interpolace, Newtonova interpolace, iterovaná interpolace, Hermitovská interpolace, matematický software *R*

**Počet stran:** 58

**Počet příloh:** 1

**Jazyk:** český

# BIBLIOGRAPHICAL IDENTIFICATION

**Author:** Ingrid Straussová

**Title:** Interpolation in software R

**Type of thesis:** Bachelor's

**Department:** Department of Mathematical Analysis and Application of Mathematics

**Supervisor:** RNDr. Jitka Machalová, Ph.D.

**The year of presentation:** 2017

**Abstract:** This bachelor thesis is focused on polynomial interpolation. Methods for finding an interpolation polynomial are presented in individual chapters. Each method is complemented by my own example. The work is completed with batch files created in mathematical software R. Codes in these batch files are programmed to find coefficients of interpolation polynomial and draw a graph for this interpolation polynomial.

**Key words:** interpolation, polynomial interpolation, Lagrange interpolation, Newton interpolation, iteration interpolation, Hermite interpolation, mathematical software *R*

**Number of pages:** 58

**Number of appendices:** 1

**Language:** czech

**Prohlášení**

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením paní RNDr. Jitky Machalové, Ph.D. a všechny použité zdroje jsem uvedla v seznamu literatury.

V Olomouci dne ..... .....

podpis

# Obsah

**Použité značení** **7**

**Úvod** **8**

**1. Úvod do interpolace** **9**
1.1. Metoda neurčitých koeficientů . . . . . 10

**2. Lagrangeova interpolace** **16**

**3. Newtonova interpolace** **23**

**4. Iterovaná interpolace** **30**
4.1. Nevillův algoritmus . . . . . 31
5.2. Aitkenův algoritmus . . . . . 37

**5. Hermitovská interpolace** **44**
5.1. Hermitovská interpolace metodou neurčitých koeficientů . . . . . 47

**6. Shrnutí** **52**

**Závěr** **57**

**Literatura** **58**

**Poděkování**

Ráda bych poděkovala především své vedoucí bakalářské práce RNDr. Jitce Machalové, Ph.D. za dostatek trpělivosti a cenné rady při psaní této práce. Dále bych chtěla poděkovat panu Mgr. Ondřeji Vencálkovi Ph.D. za cenné rady a pomoc při práci v *R*. Velké poděkování patří také mé rodině za podporu během studia.

# Použité značení

| | |
|---|---|
| $x_i$ | interpolační uzly |
| $f_i$ | funkční hodnoty v uzlech interpolace |
| $\prod_n$ | množina všech reálných polynomů stupně nejvýše n |
| $p_n(x)$ | polynom stupně nejvýše n |
| $l_i(x)$ | Lagrangeův fundamentální polynom |
| $f[x_i]$ | poměrná diference nultého řádu |
| $f[x_0, x_1, \ldots, x_n]$ | poměrná diference $n$-tého řádu |
| $N$ | přirozená čísla |
| $R$ | reálná čísla |
| $Z$ | celá čísla |
| $\sum, \prod$ | symboly pro součet a součin |

# Úvod

V matematice se často naskytne otázka, zda je možné proložit polynomickou křivku zadanými body v rovině. Dále se můžeme ptát, jestli lze nahradit polynomem složitou funkci, se kterou má tento polynom několik společných bodů. Odpovědi na tyto otázky nám dá polynomiální interpolace, kterou se v této práci budeme zabývat.

Cílem této bakalářské práce je seznámit čtenáře s polynomiální interpolací a představit metody, kterými je možné v numerické matematice nalézt interpolační polynom. Všechny metody budou podrobně vysvětleny na vlastních příkladech, které byly vymyšleny tak, aby se s nimi dobře počítalo.

V práci budou předvedeny jednotlivé dávkové soubory vytvořené v matematickém softwaru $R$, pomocí nichž je možné nalézt koeficienty interpolačního polynomu pro konkrétní data nebo si vykreslit pro takový interpolační polynom graf. Předpokládá se, že uživatel má základní znalosti pro práci v $R$, popř. doporučuji [5], [6] a [7].

# 1. Úvod do interpolace

V této části práce, pro kterou jsem čerpala z [2], [3] a [4], popíšu a nadefinuju polynomiální interpolaci. Dále vysvětlím metodu neurčitých koeficientů, ručně vypočítám vlastní příklad, který následně vyřeším také v softwaru R. Pomocí tohoto programu vykreslím i graf.

Interpolace je jedním z nejdůležitějších nástrojů numerické matematiky. Metodu interpolace volíme ve dvou případech:

- máme-li explicitně danou funkci $f(x)$, která je vyjádřena složitým předpisem a chceme-li ji nahradit jednodušší funkcí, tzn. chceme ji aproximovat jinou funkcí, která bude v předepsaných bodech nabývat stejných hodnot jako funkce $f(x)$,
- máme-li dány body $(x_i, f_i),\ i = 0, 1, \ldots, n$ a hledáme-li takovou funkci, která v bodech $x_i$ nabývá předepsaných hodnot $f_i$.

Funkci, kterou v těchto případech hledáme, nazýváme interpolační funkce.

V této práci se zaměřím na interpolaci, kdy je interpolační funkcí polynom. Budu se tedy zabývat polynomiální interpolací.

Předpokládejme, že jsou dány body $x_0, x_1, \ldots, x_n,\ x_i \neq x_k\ pro\ i \neq k$. Tyto body nazýváme uzly či uzlové body. Dále budeme značit $f_i = f(x_i)$.

Vysvětlíme si nyní, co je úlohou polynomiální interpolace. Nejdříve označme $\prod_n$ jako množinu všech reálných polynomů stupně nejvýše $n$ tvaru:

$$p_n(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_n, \tag{1}$$

kde $a_0, \ldots, a_n$ jsou libovolné reálné konstanty.

Naší úlohou je, abychom našli takový polynom $p_n(x) \in \prod_n$, který v $(n+1)$ vzájemně různých bodech $x_0,\ x_1,\ \ldots,\ x_n$ nabývá předepsaných hodnot $f_i,\ \ i = 0, 1, \ldots, n$. Tzn. chceme takový polynom $p_n(x)$, pro který platí:

$$p_n(x_i) = f_i \quad \forall i = 0, 1, \ldots, n. \tag{2}$$

Tento polynom je pak interpolačním polynomem a podmínky (2) nazveme podmínky interpolace.

**Věta 1** *Pro $(n+1)$ daných dvojic $(x_i, f_i),\ i = 0, 1, \ldots, n,\ x_i \neq x_k$ pro $i \neq k$ existuje právě jeden polynom $p_n(x) \in \prod_n$ takový, že*

$$p_n(x_i) = f_i \quad \forall i = 0, 1, \ldots, n \tag{3}$$

**Důkaz 1** *: viz [2], str. 158.*

**Poznámka 1** Z předchozí věty vyplývá, že existuje právě jeden interpolační polynom. Lze jej nalézt několika metodami. Je však důležité si uvědomit, že nám všechny metody najdou tentýž polynom.

Je známo, že nejčastěji používaným interpolačním polynomem je kubický polynom, protože u interpolačních polynomů vyšších stupňů vzniká oscilace a polynomiální interpolace ztrácí význam. V takových případech je výhodnější interpolace splajny, což jsou po částech polynomické funkce. V této práci se však tímto tématem nezabývám a čtenáře tudíž odkazuji na skripa [2]. Názornou ukázku oscilace polynomů stupně 6, 10 a 15 uvedu v kapitole 6.

## 1.1. Metoda neurčitých koeficientů

V případě, že nemáme žádné znalosti o problematice interpolačních polynomů, můžeme úlohu interpolace řešit pomocí soustavy lineárních rovnic. Taková metoda se nazývá metoda neurčitých koeficientů. Tento způsob však kvůli rozsáhlému výpočtu není vhodný pro větší počet uzlů. Víme, že hledáme polynom $n$-tého stupně, který má splňovat interpolační podmínky (2). Ty lze rozepsat

do tvaru:

$$\begin{aligned} p_n(x_0) &= a_0 x_0^n + a_1 x_0^{n-1} + \ldots + a_n = f_0 \\ p_n(x_1) &= a_0 x_1^n + a_1 x_1^{n-1} + \ldots + a_n = f_1 \\ &\vdots \\ p_n(x_n) &= a_0 x_n^n + a_1 x_n^{n-1} + \ldots + a_n = f_n. \end{aligned}$$

Dostali jsme soustavu $(n+1)$ lineárních rovnic o $(n+1)$ neznámých ve tvaru:

$$a_0 x_i^n + a_1 x_i^{n-1} + \ldots + a_n = f_i, \ i = 0, 1, \ldots, n. \tag{4}$$

Předpokládali jsme vzájemně různé body interpolace $x_0, \ldots, x_n$, proto je determinant této soustavy, tzv. Vandermondův determinant, nenulový a tato soustava má právě jedno řešení. Takovou soustavu řešíme např. Gaussovou eliminační metodou, viz např. [2],[1].

**Příklad 1** Najděte interpolační polynom $p_n(x)$ pro data daná tabulkou:

| $i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $x_i$ | $-1$ | 0 | 2 | 5 |
| $f_i$ | 10 | 3 | 13 | $-2$ |

*Řešení:* hledáme polynom $p_3(x)$, jelikož $n = 3$. Dle vztahu (4):

$$\begin{aligned} (-1)^3 a_0 + (-1)^2 a_1 + (-1)^1 a_2 + (-1)^0 a_3 &= 10 \\ 0^3 a_0 + 0^2 a_1 + 0^1 a_2 + 0^0 a_3 &= 3 \\ 2^3 a_0 + 2^2 a_1 + 2^1 a_2 + 2^0 a_3 &= 13 \\ 5^3 a_0 + 5^2 a_1 + 5^1 a_2 + 5^0 a_3 &= -2. \end{aligned}$$

Tuto soustavu vyřešíme např. Gaussovou eliminační metodou a dostaneme koeficienty $a_0$ až $a_3$. Tj. $a_0 = -1$, $a_1 = 5$, $a_2 = -1$, $a_3 = 3$. Interpolační polynom je tedy ve tvaru $p_3(x) = -x^3 + 5x^2 - x + 3$ (viz Obrázek 1). Ověříme podmínky

interpolace tak, že dosadíme předepsané body z tabulky do interpolačního polynomu.

$$-(-1)^3 + 5(-1)^2 - (-1) + 3 = 10$$
$$-(0)^3 + 5(0)^2 - (0) + 3 = 3$$
$$-(2)^3 + 5(2)^2 - (2) + 3 = 13$$
$$-(5)^3 + 5(5)^2 - (5) + 3 = -2.$$

Tímto jsme si ověřili, že jsou splněny interpolační podmínky. Řešení bylo tedy správné.

V programu $R$ jsem pro nalezení interpolačního polynomu metodou neurčitých koeficientů vytvořila program, který jsem nazvala **MNK-kalkulator.R** . Program obsahuje dva soubory:

- **mnk-koef.R**
- **graf-interpol-poly.R**

První soubor je obecný kód pro výpočet koeficientů interpolačního polynomu, seřazených od $a_0$ až po $a_n$ dle vztahu (1). Druhým souborem je kód pro vykreslení grafu interpolačního polynomu. Tento kód je analogický i pro ostatní metody. Čtenář tento program najde na přiloženém CD.

Uživatel zadá pouze vstup do části *zadejte data*. Zde jsou přichystané dva vektory, kam vloží hodnoty $x_i$ a $f_i$. Poté pomocí příkazu *source* přikáže spustit oba soubory. $R$ mu jako výstup vypíše vektor koeficientů a vykreslí graf.

Ukážeme si tento postup konkrétně na Příkladu 1.

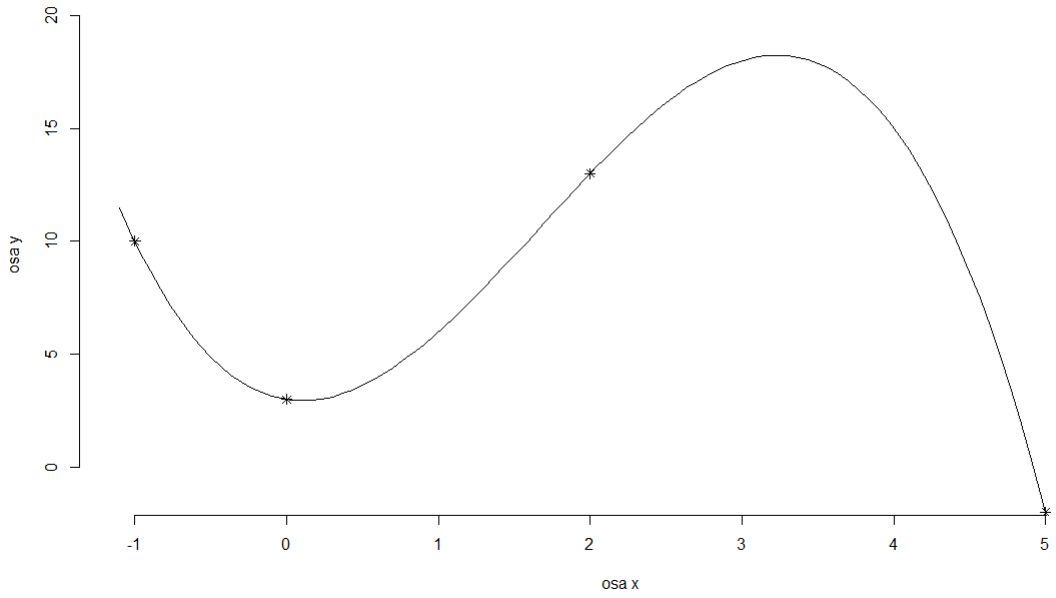
Program **MNK-kalkulator.R** vypadá takto:

```
#METODA NEURČITÝCH KOEFICIENTŮ
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
```

```
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi <- c(-1,0,2,5)
fi <- c(10,3,13,-2)
### spustte soubor pro hledani koeficientu:
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("mnk-koef.R")
### pote spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

Jako výstup dostaneme vektor koeficientů a obrázek, viz Obrázek 1, na kterém je vykreslen graf.

```
> source("mnk-koef.R")
[1] -1  5 -1  3
```




Obrázek 1: Graf interpolačního polynomu určeného metodou neurčitých koeficientů pro data z Příkladu 1

Zde uvádím náhled na kódy uvedených souborů:

- soubor **mnk-koef.R**:

```
# METODA NEURČITÝCH KOEFICIENTŮ
# vypocet koeficientu pro interpolacni polynom stupně n-1
### overeni podminky:
n <- length(xi)
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
### pomocne promenne:
x <- seq(from=min(xi)-0.1*abs(min(xi)),
          to=max(xi)+0.1*abs(max(xi)),length=100)
### definice M:
M <- matrix(nrow=n,ncol=n)
for(j in 1:n)
```

```
{M[,j] <- xi^(n-j)}
### vypocet koeficientu:
p <- solve(M,fi)
print(k <- as.numeric(p))
```

- soubor **graf-interpol-poly.R**:

```
# GRAF
# interpolacni polynom stupne n-1
if(n==1)
  {P <- k[1]*(x^(n-1));
  horni <- fi[1]+0.1*abs(max(fi));
  dolni <- fi[1]-0.1*abs(min(fi));
  plot(x,P,xlab="osa x",ylab="osa y",type="l",
       ylim=c(dolni,horni),
       frame=FALSE);
  points(fi~xi,pch=8)}
if(n!=1)
  {P <- k[1]*(x^(n-1));
  for(i in 2:n)
  {P <- (P + k[i]*(x^(n-i)))};
  uvnitr <- ((x>=min(xi))&(x<=max(xi)));
  r = range(P[uvnitr]);
  horni = r[2]+0.1*abs(r[2]);
  dolni = r[1]-0.1*abs(r[1]);
  plot(x,P,xlab="osa x",ylab="osa y",type="l",
       ylim=c(dolni,horni),
       frame=FALSE);
  points(fi~xi,pch=8)}
```

# 2. Lagrangeova interpolace

V této kapitole jsem čerpala z [2], [3]. Budu se zde zabývat Lagrangeovou metodou, která patří k nejznámějším interpolačním metodám. I tady ručně vypočítám vlastní příklad, který pak vyřeším také v softwaru R. Pomocí tohoto programu vykreslím i graf.

Ukážeme si nyní, jak lze interpolační polynom sestrojit v Lagrangeově tvaru.

**Definice 1** *Pro daná data* $(x_i, f_i)$, $i = 0, 1, \ldots, n$, $x_i \neq x_k$ *pro* $i \neq k$ *definujeme Lagrangeův tvar interpolačního polynomu vztahem:*

$$p_n(x) = l_0(x)f(x_0) + l_1(x)f(x_1) + \ldots + l_n(x)f(x_n), \tag{5}$$

*kde* $l_i(x)$ *je* $i$*-tý fundamentální polynom* $n$*-tého stupně a je dán vztahem:*

$$l_i(x) = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}. \tag{6}$$

Lehce se ověří, že tento Lagrangeův interpolační polynom (5) splňuje interpolační podmínky (2). Navíc, protože je lineární kombinací polynomů $l_i(x)$ stupně $n$, je Lagrangeův polynom $p_n(x)$ stupně nejvýše $n$.

**Příklad 2** Najděte interpolační polynom $p_n(x)$ pro data daná tabulkou:

| $i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $x_i$ | $-1$ | 0 | 2 | 5 |
| $f_i$ | $-4$ | 1 | $-1$ | 146 |

*Řešení:* hledáme polynom $p_3(x)$, jelikož $n = 3$.
Nejprve si spočítáme fundamentální polynomy pro $i = 0, 1, 2, 3$ dle vztahu (6).

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-0)(x-2)(x-5)}{(-1-0)(-1-2)(-1-5)} =$$
$$= -\frac{1}{18}(x^3 - 7x^2 + 10x),$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-(-1))(x-2)(x-5)}{(0-(-1))(0-2)(0-5)} =$$
$$= \frac{1}{10}(x^3 - 6x^2 + 3x + 10),$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-(-1))(x-0)(x-5)}{(2-(-1))(2-0)(2-5)} =$$
$$= -\frac{1}{18}(x^3 - 4x^2 - 5x),$$

$$l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-(-1))(x-0)(x-2)}{(5-(-1))(5-0)(5-2)} =$$
$$= \frac{1}{90}(x^3 - x^2 - 2x).$$

Dle vztahu (5):

$$p_3(x) = l_0(x)f_0 + l_1(x)f_1 + l_2(x)f_2 + l_3(x)f_3,$$

pro daná data dostáváme Lagrangeův interpolační polynom:

$$p_3(x) = \left(-\frac{1}{18}\right)(x^3 - 7x^2 + 10x)\,(-4) + \frac{1}{10}\,(x^3 - 6x^2 + 3x + 10)\,(1) +$$
$$+ \left(-\frac{1}{18}\right)(x^3 - 4x^2 - 5x)\,(-1) + \frac{1}{90}\,(x^3 - x^2 - 2x)\,(146) = \ldots$$
$$= 2x^3 - 4x^2 - x + 1$$

Tento polynom je vykreslen na Obrázku 2.

Lagrangeův interpolační polynom je základem pro odvození metod numerického derivování a integrování. Je tedy po teoretické stránce v numerické matematice velmi důležitý. V případě změny počtu uzlů je nutné přepočítat všechny fundamentální polynomy. Výpočty jsou proto velmi zdlouhavé a časově náročné. V tomto případě nebo při velkém počtu uzlů se doporučuje použít spíše některou z metod, které jsou uvedeny v dalších kapitolách.

V *R* jsem vytvořila program **LAGRANGE-kalkulator.R**, který obsahuje soubory s kódy pro výpočet koeficientů interpolačního polynomu. Jak již bylo uvedeno, doporučuje se interpolovat polynomem lineárním, kvadratickým či kubickým. Z toho důvodu jsem vytvořila kódy pro nalezení interpolačního polynomu stupně $0, 1, 2, 3$. Dále i stupně 4 a 5, protože zde již můžeme zaznamenat oscilaci. Jedná se o tyto soubory:

- **lagrange-koef-0st.R**
- **lagrange-koef-1st.R**
- **lagrange-koef-2st.R**
- **lagrange-koef-3st.R**
- **lagrange-koef-4st.R**
- **lagrange-koef-5st.R**

Dále program obsahuje kód funkce pro vykreslení grafu interpolačního polynomu **graf-interpol-poly.R**. Čtenář tento program najde na přiloženém CD.

Uživatel postupuje tak, že nejprve zadá vstup do části *zadejte data*. Zde jsou přichystané dva vektory, kam vloží hodnoty $x_i$ a $f_i$. Pak dle počtu zadaných dat vybere soubor, který použije pro výpočet koeficientů interpolačního polynomu a pomocí příkazu *source* přikáže spustit tento soubor. Dále pomocí příkazu *source* přikáže spustit i soubor pro vykreslení grafu. *R* jako výstup vypíše vektor koeficientů a vykreslí graf.

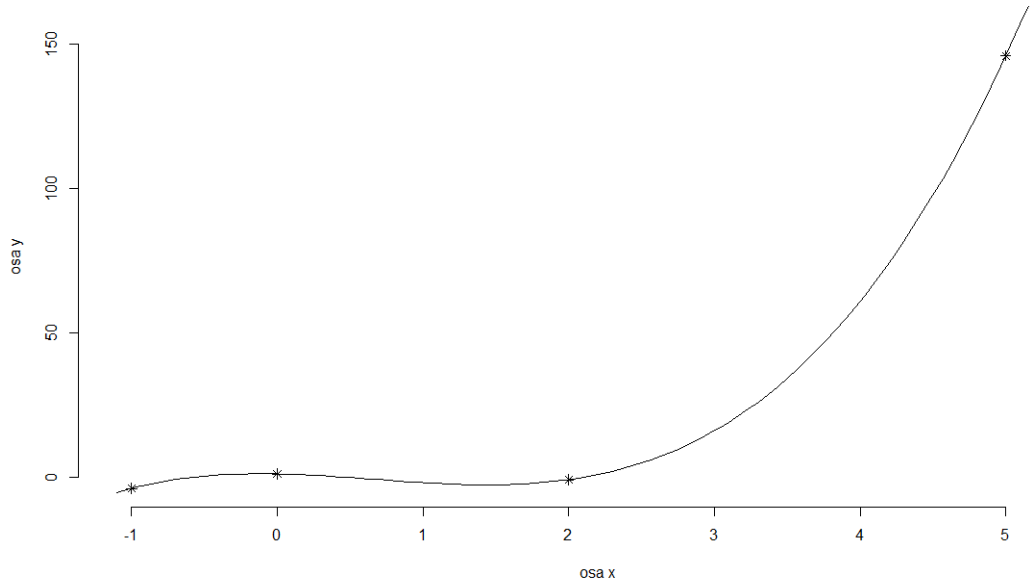
Ukážeme si tento postup konkrétně na Příkladu 2.

Program **LAGRANGE-kalkulator.R** vypadá takto:

```
# LAGRANGE
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi <- c(-1,0,2,5)
fi <- c(-4,1,-1,146)
### dle poctu vstupnich dat zvolte a spustte soubor pro hledani
### koeficientu:
### ( pro n vstupnich dat xi zvolte "lagrange-koef-(n-1)st.R")
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("lagrange-koef-0st.R")
source("lagrange-koef-1st.R")
source("lagrange-koef-2st.R")
source("lagrange-koef-3st.R")
source("lagrange-koef-4st.R")
source("lagrange-koef-5st.R")
### spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

V Příkladu 2 máme čtyři vstupní data, budeme tedy hledat interpolační polynom stupně 3. Spustíme proto soubor **lagrange-koef-3st.R** pro nalezení koeficientů interpolačního polynomu stupně 3. Poté spustíme soubor **graf-interpol-poly.R** pro vykreslení grafu. Jako výstup dostaneme vektor koeficientů a obrázek, viz Obrázek 2, na kterém je vykreslen graf.

```
> source("lagrange-koef-3st.R")
[1]  2 -4 -1  1
```




Obrázek 2: Graf interpolačního polynomu určeného Lagrangeovou metodou pro data z Příkladu 2

Na ukázku zde uvádím kód ze souboru **lagrange-koef-3st.R**, který jsem použila ke hledání interpolačního polynomu z Příkladu 2. Obecný kód pro vykreslení grafu hledaného interpolačního polynomu je analogický jako u metody neurčitých koeficientů. Zbývající kódy jsou k dispozici na CD.

```
# LAGRANGE
# vypocet koeficientu pro interpolacni polynom stupně 3
### overeni podminky:
n <- length(xi)
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
### pomocne promenne:
x <- seq(from=min(xi)-0.1*abs(min(xi)),
         to=max(xi)+0.1*abs(max(xi)),length=100)
l = length(x)
### pocatecni nulová matice L:
```

```
L <- matrix(0, n, n, byrow = TRUE)
### fundamentální polynomy:
#jmenovatel stejny, a = koeficient u x s (n-1)-ní mocninou
## l1 * fi[1] -> L[,1]
# citatel u a = konstanta
a <- (  1 / ((xi[1]-xi[2]) * (xi[1]-xi[3]) * (xi[1]-xi[4])))
# citatel u b = minus soucet jednotlivcu
b <- ( - (  xi[2] + xi[3] + xi[4])
       / ((xi[1]-xi[2]) * (xi[1]-xi[3]) * (xi[1]-xi[4])))
# citatel u c = soucet soucinu dvojic
c <- ((  xi[2] * xi[3] + xi[2] * xi[4] + xi[3] * xi[4])
       / ((xi[1]-xi[2]) * (xi[1]-xi[3]) * (xi[1]-xi[4])))
# citatel u d = minus soucet soucinu trojic
d <- ( - (xi[2] * xi[3] * xi[4])
       / ((xi[1]-xi[2]) * (xi[1]-xi[3]) * (xi[1]-xi[4])))
L[,1] <- c(a,b,c,d) * fi[1]
## l2 * fi[2] -> L[,2]
# citatel u a = konstanta
a <- (  1 / ((xi[2]-xi[1]) * (xi[2]-xi[3]) * (xi[2]-xi[4])))
# citatel u b = minus soucet jednotlivcu
b <- (- (  xi[1] + xi[3] + xi[4])
      / ((xi[2]-xi[1]) * (xi[2]-xi[3]) * (xi[2]-xi[4])))
# citatel u c = soucet soucinu dvojic
c <- ( (  xi[1]*xi[3] + xi[1]*xi[4] + xi[3]*xi[4])
      / ((xi[2]-xi[1]) * (xi[2]-xi[3]) * (xi[2]-xi[4])))
# citatel u d = minus soucet soucinu trojic
d <- (- (xi[1]*xi[3]*xi[4])
      / ((xi[2]-xi[1]) * (xi[2]-xi[3]) * (xi[2]-xi[4])))
L[,2] <- c(a,b,c,d) * fi[2]
## l3 * fi[3] -> L[,3]
```

```
# citatel u a = konstanta
a <- (  1 / ((xi[3]-xi[1]) * (xi[3]-xi[2]) * (xi[3]-xi[4])))
# citatel u b = minus soucet jednotlivcu
b <- (- (  xi[1] + xi[2] + xi[4])
      / ((xi[3]-xi[1]) * (xi[3]-xi[2]) * (xi[3]-xi[4])))
# citatel u c = soucet soucinu dvojic
c <- ( (  xi[1]*xi[2] + xi[1]*xi[4] + xi[2]*xi[4])
      / ((xi[3]-xi[1]) * (xi[3]-xi[2]) * (xi[3]-xi[4])))
# citatel u d = minus soucet soucinu trojic
d <- (- (xi[1]*xi[2]*xi[4])
      / ((xi[3]-xi[1]) * (xi[3]-xi[2]) * (xi[3]-xi[4])))
L[,3] <- c(a,b,c,d) * fi[3]
## l4 * fi[4] -> L[,4]
# citatel u a = konstanta
a <- (  1  / ((xi[4]-xi[1]) * (xi[4]-xi[2]) * (xi[4]-xi[3])))
# citatel u b = minus soucet jednotlivcu
b <- (- (  xi[1] + xi[2] + xi[3])
      / ((xi[4]-xi[1]) * (xi[4]-xi[2]) * (xi[4]-xi[3])))
# citatel u c = soucet soucinu dvojic
c <- ( (  xi[1]*xi[2] + xi[1]*xi[3] + xi[2]*xi[3])
      / ((xi[4]-xi[1]) * (xi[4]-xi[2]) * (xi[4]-xi[3])))
# citatel u d = minus soucet soucinu trojic
d <- (- (xi[1]*xi[2]*xi[3])
      / ((xi[4]-xi[1]) * (xi[4]-xi[2]) * (xi[4]-xi[3])))
L[,4] <- c(a,b,c,d) * fi[4]
### koeficienty:
k <- c(1:n)
for (j in 1:n)
{k[j]  <- sum(L[j,]) }
print(k)
```

# 3. Newtonova interpolace

V této kapitole se budu věnovat Newtonově interpolaci, která je velmi užitečná při praktických výpočtech. Zde jsem čerpala z [2], [3] a [4]. Ukážu opět vlastní příklad, který vyřeším ručně i v $R$, a také vykreslím pro daná data graf.

Hledáme-li interpolační polynom Newtonovou metodou a zvýšíme-li počet uzlů, některé vypočtené hodnoty se nezmění a můžeme je znovu použít. Newtonova metoda je proto v porovnání např. s Lagrangeovou metodou výhodnější pro výpočty. U Lagrangeovy metody je nutné při změně počtu uzlů přepočítat všechny fundamentální polynomy, což není vhodné pro ruční počítání. Při hledání Newtonova interpolačního polynomu využíváme poměrné diference.

Mějme dány navzájem různé body $x_0, \ldots, x_n$ a funkční hodnoty $f_0, \ldots, f_n$ v těchto bodech.

**Definice 2** *[3] Nechť je dána množina bodů* $\{x_i, \quad i \in Z\}$, $x_i \neq x_j$ *pro* $i \neq j$, $i, j \in Z$ *a nechť funkce* $f(x)$ *je definovaná v těchto bodech. Pak poměrnou diferenci 1. řádu, resp. první poměrnou diferenci funkce* $f(x)$ *v bodech* $x_i, x_j$ *definujeme vztahem:*

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i} \quad pro\ i \neq j.$$

*Poměrnou diferenci k-tého řádu, resp. k-tou poměrnou diferenci funkce* $f(x)$ *v bodech* $x_i, x_{i+1}, \ldots, x_{i+k}$ *definujeme pro* $k \in N, k \geq 2$ *vztahem:*

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}. \qquad (7)$$

**Věta 2** *Newtonův interpolační polynom* $p_n(x) \in \prod_n$ *pro body* $(x_i, f(x_i))$, *kde* $x_i \neq x_k$ *pro* $i \neq k$, $i = 0, 1, \ldots n$, *je tvaru:*

$$p_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + ... + f[x_0, ..., x_n](x - x_0)...(x - x_{n-1}) \quad (8)$$

**Důkaz 2** *: viz [2], str. 163.*

**Příklad 3** Sestrojte Newtonův interpolační polynom $p_n(x)$ pro data, která jsou dána tabulkou:

| $x_i$ | $-3$ | $-2$ | $-1$ | $0$ | $1$ | $2$ |
|---|---|---|---|---|---|---|
| $f_i$ | 650 | 103 | 4 | $-1$ | $-2$ | $-65$ |

*Řešení:* hledáme polynom $p_5(x)$, jelikož $n = 5$.
Sestrojíme tabulku poměrných diferencí. Hodnoty diferencí vypočítáme dle vzorce (7). Z důvodu úspory místa budeme v tabulce značit $f[x_i, \ldots, x_{i+k}] = f[_{i,\ldots,i+k}]$.

| $i$ | $x_i$ | $f_i$ | $f[_{i,i+1}]$ | $f[_{i,i+1,i+2}]$ | $f[_{i,i+1,i+2,i+3}]$ | $f[_{i,i+1,\ldots,i+4}]$ | $f[_{i,i+1,\ldots,i+5}]$ |
|---|---|---|---|---|---|---|---|
| 0 | $-3$ | 650 | $-547$ | 224 | $-59$ | 11 | $-2$ |
| 1 | $-2$ | 103 | $-99$ | 47 | $-15$ | 1 | |
| 2 | $-1$ | 4 | $-5$ | 2 | $-11$ | | |
| 3 | 0 | $-1$ | $-1$ | $-31$ | | | |
| 4 | 1 | $-2$ | $-63$ | | | | |
| 5 | 2 | $-65$ | | | | | |

Pro výpočet využijeme pouze poměrné diference z prvního řádku tabulky. Dle vztahu (8):

$$\begin{aligned} p_5(x) = {} & f(x_0) + \\ & + (x - x_0) f[x_0, x_1] + \\ & + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \\ & + (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3] + \\ & + (x - x_0)(x - x_1)(x - x_2)(x - x_3) f[x_0, x_1, x_2, x_3, x_4] + \\ & + (x - x_0)(x - x_1)(x - x_2)(x - x_3)(x - x_4) f[x_0, x_1, x_2, x_3, x_4, x_5] \end{aligned}$$

Dosadíme:

$$\begin{aligned}
p_5(x) = 650 &+ \\
&+ (x-(-3))\,(-547) + \\
&+ (x-(-3))\,(x-(-2))\,224 + \\
&+ (x-(-3))\,(x-(-2))\,(x-(-1))\,(-59) \; + \\
&+ (x-(-3))\,(x-(-2))\,(x-(-1))\,(x-0)\,(11) \; + \\
&+ (x-(-3))\,(x-(-2))\,(x-(-1))\,(x-0)\,(x-1)\,(-2) \; = \\
&= 650 - 547x - 1641 + 224x^2 + 1120x + 1344 - 59x^3 - 354x^2 - 649x \\
&\quad - 354 + 11x^4 + 66x^3 + 121x^2 + 66x - 2x^5 - 10x^4 - 10x^3 + 10x^2 + 12x = \\
&= -2x^5 + x^4 - 3x^3 + x^2 + 2x - 1.
\end{aligned}$$

Tento interpolační polynom je vykreslen na Obrázku 3.

V *R* jsem vytvořila program **NEWTON-kalkulator.R**, který obsahuje soubory s kódy pro výpočet koeficientů interpolačního polynomu. Analogicky jako u Lagrangeovy metody jsem vytvořila kódy pro nalezení interpolačního polynomu stupně $0, 1, 2, 3, 4$ a 5. Jedná se o tyto soubory:

- **newton-koef-0st.R**
- **newton-koef-1st.R**
- **newton-koef-2st.R**
- **newton-koef-3st.R**
- **newton-koef-4st.R**
- **newton-koef-5st.R**

I tento program obsahuje kód funkce pro vykreslení grafu interpolačního polynomu **graf-interpol-poly.R**. Čtenář tento program najde na přiloženém CD.

Uživatel postupuje stejně jako u Lagrangeovy interpolace. Nejprve zadá vstup do části *zadejte data*. Zde jsou přichystané dva vektory, kam vloží hodnoty $x_i$ a

$f_i$. Dle počtu zadaných dat vybere soubor, který použije pro výpočet koeficientů interpolačního polynomu a pomocí příkazu *source* přikáže spustit tento soubor. Následně pomocí příkazu *source* přikáže spustit i soubor pro vykreslení grafu. $R$ jako výstup vypíše vektor koeficientů a vykreslí graf.

Ukážeme si tento postup konkrétně na Příkladu 3.

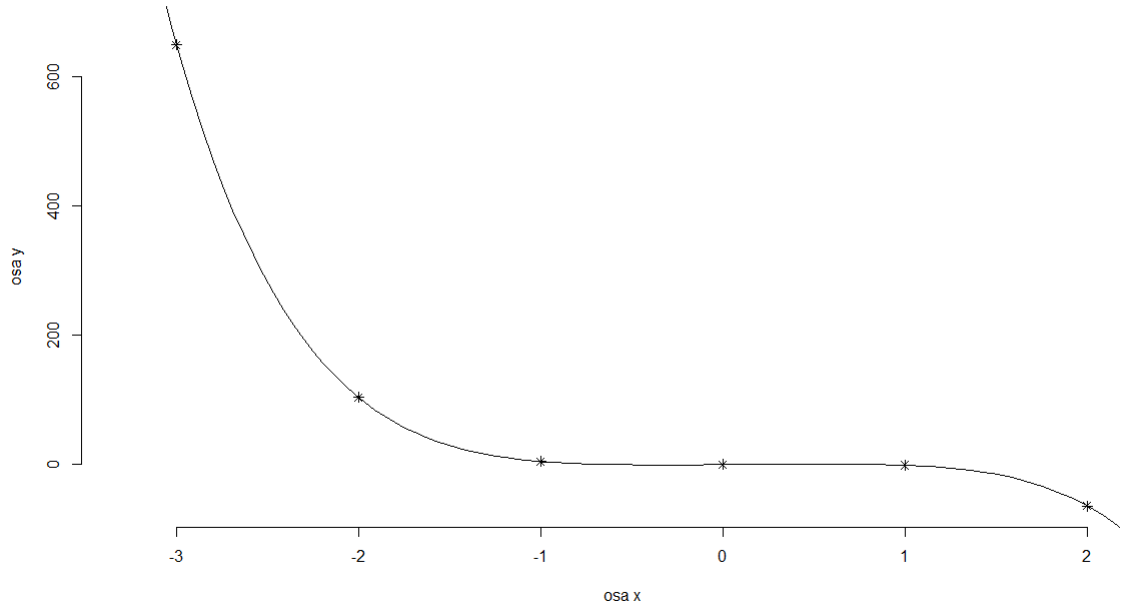
Program **NEWTON-kalkulator.R** vypadá takto:

```
# NEWTON
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi <- c(-3,-2,-1,0,1,2)
fi <- c(650,103,4,-1,-2,-65)
### dle poctu vstupnich dat zvolte a spustte soubor pro hledani
### koeficientu:
### ( pro n vstupnich dat xi zvolte "newton-koef-(n-1)st.R")
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("newton-koef-0st.R")
source("newton-koef-1st.R")
source("newton-koef-2st.R")
source("newton-koef-3st.R")
source("newton-koef-4st.R")
source("newton-koef-5st.R")
### spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

V Příkladu 3 máme šest vstupních dat, budeme tedy hledat interpolační polynom stupně 5. Spustíme proto soubor **newton-koef-5st.R** pro nalezení koefi-

cientů interpolačního polynomu stupně 5. Poté spustíme soubor **graf-interpol-poly.R** pro vykreslení grafu. Jako výstup dostaneme vektor koeficientů a obrázek, viz Obrázek 3, na kterém je vykreslen graf.

```
> source("newton-koef-5st.R")
[1] -2  1 -3  1  2 -1
```




Obrázek 3: Graf interpolačního polynomu určeného Newtonovou metodou pro data z Příkladu 3

Na ukázku zde uvádím kód ze souboru **newton-koef-5st.R**, který jsem použila ke hledání interpolačního polynomu z Příkladu 3. Obecný kód pro vykreslení grafu hledaného interpolačního polynomu je analogický jako u metody neurčitých koeficientů. Zbývající kódy jsou k dispozici na CD.

```
# NEWTON
# vypocet koeficientu pro interpolacni polynom stupně 5
### overeni podminky:
n <- length(xi)
```

```
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
### pomocne promenne:
x <- seq(from=min(xi)-1,to=max(xi)+1,length=100)
### pocatecni nulová matice D:
D <- matrix(0, n, n, byrow = TRUE)
### matice diferenci D:
d <- fi[1:n]
D[1:n,1] <- d
for (i in 0:(n-2))
{d <- (d[2:(n-i)]-d[1:((n-i)-1)])
      /(xi[(i+2):n]-xi[1:((n-i)-1)])
D[1:(n-(i+1)),i+2] <- d   }
### koeficienty:
# k[1] - koeficinet pro x s nejvyšší mocninou,
# k[6] - koeficinet pro x s nejnižší mocninou
k <- c(1:n)
k[1] <-    (D[1,6])
k[2] <-    (D[1,5]
          - D[1,6] * ( xi[1]+xi[2]+xi[3]+xi[4]+xi[5]))
k[3] <-    (D[1,4]
          - D[1,5] * ( xi[1]+xi[2]+xi[3]+xi[4])
          + D[1,6] * ( xi[1]*xi[2]+xi[1]*xi[3]
                      +xi[1]*xi[4]+xi[1]*xi[5]
                      +xi[2]*xi[3]+xi[2]*xi[4]
                      +xi[2]*xi[5]+xi[3]*xi[4]
                      +xi[3]*xi[5]+xi[4]*xi[5]))
k[4] <-    (D[1,3]
          - D[1,4] * ( xi[1]+xi[2]+xi[3])
          + D[1,5] * ( xi[1]*xi[2]
```

```
                    +xi[1]*xi[3]
                    +xi[1]*xi[4]
                    +xi[2]*xi[3]
                    +xi[2]*xi[4]
                    +xi[3]*xi[4])
          - D[1,6] * ( xi[1]*xi[2]*xi[3] + xi[1]*xi[2]*xi[4]
                    +xi[1]*xi[2]*xi[5] + xi[1]*xi[3]*xi[4]
                    +xi[1]*xi[3]*xi[5] + xi[1]*xi[4]*xi[5]
                    +xi[2]*xi[3]*xi[4] + xi[2]*xi[3]*xi[5]
                    +xi[2]*xi[4]*xi[5] + xi[3]*xi[4]*xi[5]))
k[5] <-    (D[1,2]
          - D[1,3] * ( xi[1] + xi[2])
          + D[1,4] * ( xi[1]*xi[2] + xi[1]*xi[3] + xi[2]*xi[3])
          - D[1,5] * ( xi[1]*xi[2]*xi[3] + xi[1]*xi[2]*xi[4]
                    +xi[1]*xi[3]*xi[4] + xi[2]*xi[3]*xi[4])
          + D[1,6] * ( xi[1]*xi[2]*xi[3]*xi[4]
                    +xi[1]*xi[2]*xi[3]*xi[5]
                    +xi[1]*xi[2]*xi[4]*xi[5]
                    +xi[1]*xi[3]*xi[4]*xi[5]
                    +xi[2]*xi[3]*xi[4]*xi[5]))
k[6] <-    (D[1,1]
          - D[1,2] * (xi[1])
          + D[1,3] * (xi[1]*xi[2])
          - D[1,4] * (xi[1]*xi[2]*xi[3])
          + D[1,5] * (xi[1]*xi[2]*xi[3]*xi[4])
          - D[1,6] * (xi[1]*xi[2]*xi[3]*xi[4]*xi[5]))
print(k)
```

# 4. Iterovaná interpolace

V této části práce, pro kterou jsem čerpala z [2] a [4], popíšu a nadefinuju iterovanou interpolaci. Následně uvedu vlastní příklad, který vypočítám uvedenými algoritmy. Tento příklad také vyřeším v $R$ a vykreslím pro daná data graf.

Postup řešení u iterované interpolace spočívá ve výpočtu interpolačního polynomu po částech, tzv. iterovaně. To znamená, že začneme s výpočtem interpolačního polynomu pro menší počet uzlů a postupně přidáváme další, přičemž tak postupně zkonstruujeme celý interpolační polynom.

Jak získat interpolační polynom pomocí iterované interpolace nám říká následující věta.

**Věta 3** *Nechť jsou dány body* $(x_i,\ f_i)$, *kde* $x_i \neq x_k$ *pro* $i \neq k$, $i = 0, 1, \ldots, n$. *Označme* $p_{i_0 i_1 \ldots i_k}(x)$ *takový interpolační polynom z množiny* $\prod_k,\ k \leq n, k \in N$, *pro který platí:*

$$p_{i_0 i_1 \ldots i_k}(x_{i_j}) = f_{i_j}, \quad j = 0, 1, \ldots, k.$$

*Potom iterovanou interpolací zkonstruujeme interpolační polynom takto:*

$$p_{i_j}(x) = f_{i_j} \tag{9}$$

$$p_{i_0 \ldots i_k}(x) = \frac{1}{x_{i_k} - x_{i_0}} \begin{vmatrix} p_{i_1 \ldots i_k}(x) & x - x_{i_k} \\ p_{i_0 \ldots i_{k-1}}(x) & x - x_{i_0} \end{vmatrix} \tag{10}$$

**Důkaz 3** *: viz [2], str. 180.*

**Poznámka 2** $p_{i_0 \ldots i_k}(x)$ interpretujeme jako interpolační polynom pro body $x_{i_0}, \ldots, x_{i_k}$. To znamená např. $p_{23}(x)$ je interpolační polynom v bodech $x_2, x_3$.

Vzorec (10) lze psát ve tvaru:

$$p_{i_0\dots i_k}(x) = \frac{(x - x_{i_0})p_{i_1\dots i_k}(x) - (x - x_{i_k})p_{i_0\dots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}.$$

Dle vzorců z Věty 3 lze iterovanou interpolaci počítat různě. Já se budu zabývat konkrétně Nevillovým a Aitkenovým algoritmem.

## 4.1. Nevillův algoritmus

Konstrukci interpolačního polynomu dle Nevillova algoritmu provádíme tím způsobem, že postupně počítáme jednotlivé interpolační polynomy nižších stupňů podle vztahu (9) a (10), prostřednictvím nichž se propočítáme až k hledanému interpolačnímu polynomu. Pro lepší orientaci ve výpočtech si můžeme vytvořit tabulku pro tento algoritmus.

| $i$ | $x_i$ | $f_i$ | $p_{i,i+1}(x)$ | $p_{i,i+1,i+2}(x)$ | $p_{i,i+1,\dots,i+3}(x)$ | $p_{i,i+1,\dots,i+4}(x)$ |
|---|---|---|---|---|---|---|
| 0 | $x_0$ | $f_0 = p_0(x)$ | | | | |
| 1 | $x_1$ | $f_1 = p_1(x)$ | $p_{01}(x)$ | | | |
| 2 | $x_2$ | $f_2 = p_2(x)$ | $p_{12}(x)$ | $p_{012}(x)$ | | |
| 3 | $x_3$ | $f_3 = p_3(x)$ | $p_{23}(x)$ | $p_{123}(x)$ | $p_{0123}(x)$ | |
| 4 | $x_4$ | $f_4 = p_4(x)$ | $p_{34}(x)$ | $p_{234}(x)$ | $p_{1234}(x)$ | $p_{01234}(x)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

Dle vztahu (10) bude např. vzorec pro interpolační polynom $p_{01234}$ čtvrtého stupně vypadat takto:

$$p_{01234}(x) = \frac{(x - x_0)p_{1234}(x) - (x - x_4)p_{0123}(x)}{x_4 - x_0}.$$

**Příklad 4** Užitím Nevillova algoritmu najděte interpolační polynom pro daná data:

| $i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $x_i$ | $-2$ | $-1$ | 0 | 1 |
| $f_i$ | 35 | 6 | $-3$ | $-10$ |

*Řešení:* hledáme polynom $p_3(x)$, jelikož $n = 3$.
Nevillův algoritmus počítá polynomy v tabulce:

| $i$ | $x_i$ | $f_i$ | $p_{i,i+1}(x)$ | $p_{i,i+1,i+2}(x)$ | $p_{i,i+1,\ldots,i+3}(x)$ |
|---|---|---|---|---|---|
| 0 | $x_0 = -2$ | $p_0(x) = 35$ | | | |
| 1 | $x_1 = -1$ | $p_1(x) = 6$ | $p_{01}(x)$ | | |
| 2 | $x_2 = 0$ | $p_2(x) = -3$ | $p_{12}(x)$ | $p_{012}(x)$ | |
| 3 | $x_3 = 1$ | $p_3(x) = -10$ | $p_{23}(x)$ | $p_{123}(x)$ | $p_{0123}(x)$ |

Dosadíme do vztahů (9) a (10) a zkonstruujeme všechny interpolační polynomy z tabulky. Nejprve vypočítáme lineární interpolační polynomy:

$$p_{01}(x) = \frac{1}{x_1 - x_0} \begin{vmatrix} p_1(x) & x - x_1 \\ p_0(x) & x - x_0 \end{vmatrix} = \frac{1}{-1 - (-2)} \begin{vmatrix} 6 & x - (-1) \\ 35 & x - (-2) \end{vmatrix} =$$
$$= -23 - 29x,$$

$$p_{12}(x) = \frac{1}{x_2 - x_1} \begin{vmatrix} p_2(x) & x - x_2 \\ p_1(x) & x - x_1 \end{vmatrix} = \frac{1}{0 - (-1)} \begin{vmatrix} -3 & x - 0 \\ 6 & x - (-1) \end{vmatrix} =$$
$$= -9x - 3,$$

$$p_{23}(x) = \frac{1}{x_3 - x_2} \begin{vmatrix} p_3(x) & x - x_3 \\ p_2(x) & x - x_2 \end{vmatrix} = \frac{1}{1 - 0} \begin{vmatrix} -10 & x - 1 \\ -3 & x - 0 \end{vmatrix} =$$
$$= -7x - 3.$$

Podobně vypočítáme kvadratické interpolační polynomy:

$$p_{012}(x) = \frac{1}{x_2 - x_0} \begin{vmatrix} p_{12}(x) & x - x_2 \\ p_{01}(x) & x - x_0 \end{vmatrix} = \frac{1}{0 - (-2)} \begin{vmatrix} -9x - 3 & x - 0 \\ -23 - 29x & x - (-2) \end{vmatrix} =$$
$$= 10x^2 + x - 3,$$

$$p_{123}(x) = \frac{1}{x_3 - x_1} \begin{vmatrix} p_{23}(x) & x - x_3 \\ p_{12}(x) & x - x_1 \end{vmatrix} = \frac{1}{1 - (-1)} \begin{vmatrix} -7x - 3 & x - 1 \\ -9x - 3 & x - (-1) \end{vmatrix} =$$
$$= x^2 - 8x - 3.$$

A konečně hledaný interpolační polynom třetího stupně:

$$p_{0123}(x) = \frac{1}{x_3 - x_0} \begin{vmatrix} p_{123}(x) & x - x_3 \\ p_{012}(x) & x - x_0 \end{vmatrix} = \frac{1}{1 - (-2)} \begin{vmatrix} x^2 - 8x - 3 & x - 1 \\ 10x^2 + x - 3 & x - (-2) \end{vmatrix} =$$
$$= -3x^3 + x^2 - 5x - 3.$$

Poslední polynom $p_{0123}(x)$ je hledaným interpolačním polynomem třetího stupně pro daná data a je vykreslen na Obrázku 4.

Pro tento algoritmus jsem v *R* vytvořila program **NEVILL-kalkulator.R**, obsahující soubory s kódy pro výpočet koeficientů interpolačního polynomu. Stejně jako u Lagrangeovy a Newtonovy metody jsem vytvořila kódy pro nalezení interpolačního polynomu stupně $0, 1, 2, 3, 4$ a 5. Jedná se o tyto soubory:

- **nevill-koef-0st.R**
- **nevill-koef-1st.R**
- **nevill-koef-2st.R**
- **nevill-koef-3st.R**
- **nevill-koef-4st.R**
- **nevill-koef-5st.R**

Program opět obsahuje soubor s kódem pro vykreslení grafu interpolačního polynomu **graf-interpol-poly.R**. Čtenář najde tento program na přiloženém CD.

Uživatel postupuje stejně jako u Lagrangeovy a Newtonovy interpolace. Zadává vstup do části *zadejte data*. Zde jsou přichystané dva vektory. Vloží zde hodnoty $x_i$ a $f_i$. Dle počtu zadaných dat vybere soubor pro výpočet koeficientů interpolačního polynomu a pomocí příkazu *source* přikáže spustit tento soubor. Poté pomocí příkazu *source* přikáže spustit soubor pro vykreslení grafu. *R* jako výstup vypíše vektor koeficientů a vykreslí graf.

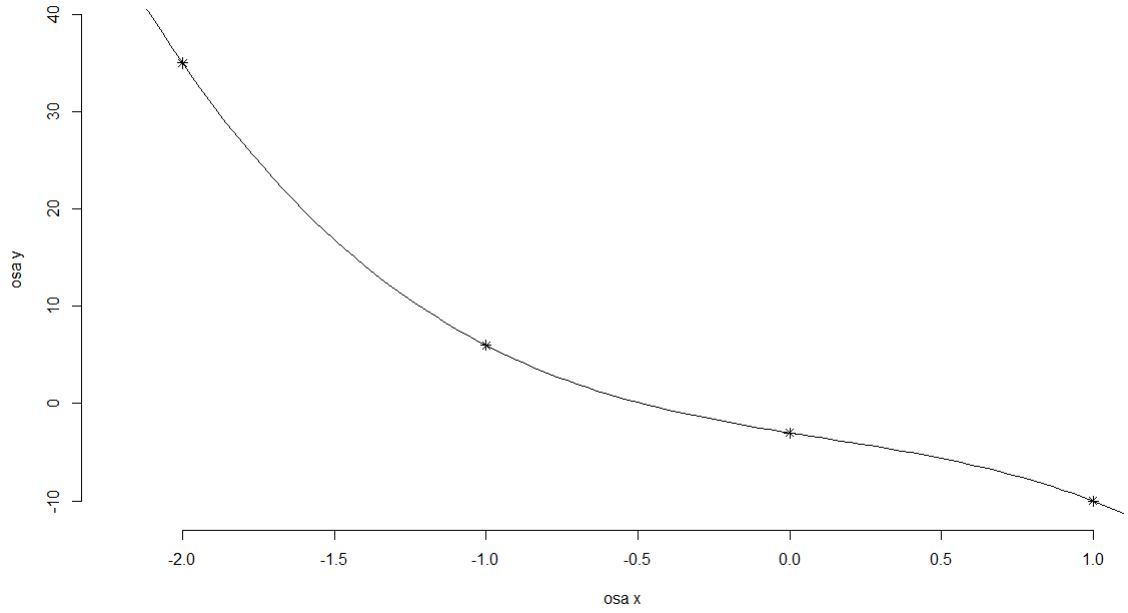
Ukážeme si tento postup konkrétně na Příkladu 4.

Program **NEVILL-kalkulator.R** vypadá takto:

```
# ITEROVANA - NEVILL
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi <- c(-2,-1,0,1)
fi <- c(35,6,-3,-10)
### dle poctu vstupnich dat zvolte a spustte soubor pro hledani
### koeficientu:
### ( pro n vstupnich dat xi zvolte "nevill-koef-(n-1)st.R")
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("nevill-koef-0st.R")
source("nevill-koef-1st.R")
source("nevill-koef-2st.R")
source("nevill-koef-3st.R")
source("nevill-koef-4st.R")
source("nevill-koef-5st.R")
### spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

V Příkladu 4 máme čtyři vstupní data, budeme tedy hledat interpolační polynom stupně 3. Spustíme soubor **nevill-koef-3st.R** pro nalezení koeficientů interpolačního polynomu stupně 3. Poté spustíme soubor **graf-interpol-poly.R** pro vykreslení grafu. Jako výstup dostaneme vektor koeficientů a obrázek, viz Obrázek 4, na kterém je vykreslen graf.

```
> source("nevill-koef-3st.R")
[1] -3  1 -5 -3
```




Obrázek 4: Graf interpolačního polynomu určeného metodou iterované interpolace pro data z Příkladu 4 a 5

Na ukázku uvádím kód ze souboru **nevill-koef-3st.R**, který jsem použila ke hledání interpolačního polynomu z Příkladu 4. Obecný kód pro vykreslení grafu hledaného interpolačního polynomu je analogický jako u ostatních metod. Zbývající kódy jsou k dispozici na CD.

```
#ITEROVANA INTERPOLACE - NEVILLUV ALGORITMUS
# vypocet koeficientu pro interpolacni polynom stupně 3
### overeni podminky:
n <- length(xi)
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
### pomocne promenne:
x <- seq(from=min(xi)-0.1*abs(min(xi)),
         to=max(xi)+0.1*abs(max(xi)),length=100)
### koeficienty postupne konstruovanych polynomu:
```

```
#P01
a <- c(1:2)
a[1] <-             (fi[2]-fi[1])/(xi[2]-xi[1])
a[2] <- (fi[1]*xi[2]-fi[2]*xi[1])/(xi[2]-xi[1])
#P12
b <- c(1:2)
b[1] <-             (fi[3]-fi[2])/(xi[3]-xi[2])
b[2] <- (fi[2]*xi[3]-fi[3]*xi[2])/(xi[3]-xi[2])
#P23
c <- c(1:2)
c[1] <-             (fi[4]-fi[3])/(xi[4]-xi[3])
c[2] <- (fi[3]*xi[4]-fi[4]*xi[3])/(xi[4]-xi[3])
#P012
f <- c(1:3)
f[1] <- (b[1]-a[1])                     /(xi[3]-xi[1])
f[2] <- (b[2]-a[2]-b[1]*xi[1]+a[1]*xi[3])/(xi[3]-xi[1])
f[3] <-          (-b[2]*xi[1]+a[2]*xi[3])/(xi[3]-xi[1])
#P123
g <- c(1:3)
g[1] <- (c[1]-b[1])                     /(xi[4]-xi[2])
g[2] <- (c[2]-b[2]-c[1]*xi[2]+b[1]*xi[4])/(xi[4]-xi[2])
g[3] <-          (-c[2]*xi[2]+b[2]*xi[4])/(xi[4]-xi[2])
#P0123
k <- c(1:4)
k[1] <- (g[1]-f[1])                     /(xi[4]-xi[1])
k[2] <- (g[2]-f[2]-g[1]*xi[1]+f[1]*xi[4])/(xi[4]-xi[1])
k[3] <- (g[3]-f[3]-g[2]*xi[1]+f[2]*xi[4])/(xi[4]-xi[1])
k[4] <-          (-g[3]*xi[1]+f[3]*xi[4])/(xi[4]-xi[1])
print(k)
```

## 5.2. Aitkenův algoritmus

Při konstrukci interpolačního polynomu dle Aitkenova algoritmu aplikujeme taktéž vztahy (9) a (10), k výpočtu však užíváme jiné polynomy. Konstruujeme postupně interpolační polynomy

$$p_0(x), p_1(x), \ldots, p_n(x),$$
$$p_{01}(x), p_{02}(x), \ldots, p_{0n}(x),$$
$$p_{012}(x), p_{013}(x), \ldots, p_{01n}(x),$$
$$\vdots$$
$$p_{0123,\ldots,n}(x).$$

Pro tento algoritmus si opět vytvoříme tabulku:

| $x_i$ | $f_i$ | $p_{0i}(x)$ | $p_{01i}(x)$ | $p_{012i}(x)$ | $p_{0123i}(x)$ |
|---|---|---|---|---|---|
| $x_0$ | $f_0 = p_0(x)$ | | | | |
| $x_1$ | $f_1 = p_1(x)$ | $p_{01}(x)$ | | | |
| $x_2$ | $f_2 = p_2(x)$ | $p_{02}(x)$ | $p_{012}(x)$ | | |
| $x_3$ | $f_3 = p_3(x)$ | $p_{03}(x)$ | $p_{013}(x)$ | $p_{0123}(x)$ | |
| $x_4$ | $f_4 = p_4(x)$ | $p_{04}(x)$ | $p_{014}(x)$ | $p_{0124}(x)$ | $p_{01234}(x)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

Dle vztahu (10) bude v tomto případě např. vzorec interpolačního polynomu $p_{014}$ pro body $x_0$, $x_1$ a $x_4$ druhého stupně vypadat takto:

$$p_{014}(x) = \frac{(x - x_1)p_{04}(x) - (x - x_4)p_{10}(x)}{x_4 - x_1}$$

Polynom $p_{04}(x)$ je interpolační polynom pro data $x_0$ a $x_4$. Podobně polynom $p_{10}(x)$ je interpolační polynom pro data $x_1$ a $x_0$. Využijeme permutace indexů a místo $p_{10}(x)$ můžeme psát $p_{01}(x)$. Jedná se totiž o stejný polynom, který interpoluje data $x_0$ a $x_1$.

Nyní si zkonstruujeme interpolační polynom pro data z Příkadu 4 pomocí Aitkenova algoritmu.

**Příklad 5** Užitím Aitkenova algoritmu najděte interpolační polynom pro daná data:

| $i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $x_i$ | $-2$ | $-1$ | 0 | 1 |
| $f_i$ | 35 | 6 | $-3$ | $-10$ |

*Řešení:* hledáme polynom $p_3(x)$, jelikož $n = 3$.

Aitkenův algoritmus počítá polynomy uvedené v tabulce:

| $i$ | $x_i$ | $f_i$ | $p_{i,i+1}(x)$ | $p_{i,i+1,i+2}(x)$ | $p_{i,i+1,\ldots,i+3}(x)$ |
|---|---|---|---|---|---|
| 0 | $x_0 = -2$ | $p_0(x) = 35$ | | | |
| 1 | $x_1 = -1$ | $p_1(x) = 6$ | $p_{01}(x)$ | | |
| 2 | $x_2 = 0$ | $p_2(x) = -3$ | $p_{02}(x)$ | $p_{012}(x)$ | |
| 3 | $x_3 = 1$ | $p_3(x) = -10$ | $p_{03}(x)$ | $p_{013}(x)$ | $p_{0123}(x)$ |

Dosadíme do vztahů (9), (10) a sestrojíme lineární interpolační polynomy $p_{01}(x), p_{02}(x), p_{03}(x)$:

$$p_{01}(x) = \frac{1}{x_1 - x_0} \begin{vmatrix} p_1(x) & x - x_1 \\ p_0(x) & x - x_0 \end{vmatrix} = \frac{1}{-1 - (-2)} \begin{vmatrix} 6 & x - (-1) \\ 35 & x - (-2) \end{vmatrix} =$$
$$= -29x - 23,$$

$$p_{02}(x) = \frac{1}{x_2 - x_0} \begin{vmatrix} p_2(x) & x - x_2 \\ p_0(x) & x - x_0 \end{vmatrix} = \frac{1}{0 - (-2)} \begin{vmatrix} -3 & x - 0 \\ 35 & x - (-2) \end{vmatrix} =$$
$$= -19x - 3,$$

$$p_{03}(x) = \frac{1}{x_3 - x_0} \begin{vmatrix} p_3(x) & x - x_3 \\ p_0(x) & x - x_0 \end{vmatrix} = \frac{1}{1 - (-2)} \begin{vmatrix} -10 & x - 1 \\ 35 & x - (-2) \end{vmatrix} =$$
$$= -15x + 5.$$

Nyní zkonstruujeme interpolační polynom $p_{012}(x)$, k jehož sestrojení bychom dle vztahu (10) potřebovali interpolační polynomy $p_{01}(x)$ a $p_{12}(x)$. Upravíme si tedy permutací indexů $p_{012}(x)$ na $p_{102}(x)$, k jehož výpočtu potřebujeme interpolační polynomy $p_{10}(x)$ a $p_{02}(x)$. Permutací indexů si upravíme $p_{10}(x)$ na $p_{01}(x)$. Nyní můžeme sestrojit interpolační polynom $p_{012}(x)$.

$$p_{012}(x) = p_{102}(x) = \frac{1}{x_2 - x_1} \begin{vmatrix} p_{02}(x) & x - x_2 \\ p_{01}(x) & x - x_1 \end{vmatrix} =$$
$$= \frac{1}{0 - (-1)} \begin{vmatrix} -19x - 3 & x - 0 \\ -23 - 29x & x - (-1) \end{vmatrix} =$$
$$= 10x^2 + x - 3.$$

Stejně tak interpolační polynom $p_{013}(x)$ upravíme permutací indexů na $p_{103}(x)$, a dále $p_{10}(x)$ na $p_{01}(x)$. Interpolační polynom $p_{013}(x)$ již můžeme zkonstruovat, protože $p_{013}(x) = p_{103}(x)$.

$$p_{013}(x) = p_{103}(x) = \frac{1}{x_3 - x_1} \begin{vmatrix} p_{03}(x) & x - x_3 \\ p_{01}(x) & x - x_1 \end{vmatrix} =$$
$$= \frac{1}{1 - (-1)} \begin{vmatrix} 5 - 15x & x - 1 \\ -23 - 29x & x - (-1) \end{vmatrix} =$$
$$= 7x^2 - 8x - 9.$$

Zbývá nám už jen sestrojit interpolační polynom $p_{0123}(x)$. Ten si permutací indexů upravíme na $p_{2013}(x)$, tudíž k jeho výpočtu použijeme $p_{201}(x)$ permutací indexů upravený na $p_{012}(x)$.

$$p_{0123}(x) = p_{2013}(x) = \frac{1}{x_3 - x_2} \begin{vmatrix} p_{013}(x) & x - x_3 \\ p_{012}(x) & x - x_2 \end{vmatrix} =$$
$$= \frac{1}{1 - 0} \begin{vmatrix} 7x^2 - 8x - 9 & x - 1 \\ 10x^2 + x - 3 & x - 0 \end{vmatrix} =$$
$$= -3x^3 + x^2 - 5x - 3.$$

Poslední polynom $p_{0123}(x)$ je hledaným interpolačním polynomem třetího stupně pro daná data a je vykreslen na Obrázku 4.

**AITKEN-kalkulator.R** je program vytvořený pro tento algoritmus. Analogicky jako **NEVILL-kalkulator.R** obsahuje soubory s kódy pro výpočet koeficientů interpolačního polynomu, které jsou i zde k dispozici pro stupeň polynomu $0, 1, 2, 3, 4$ a 5. Jedná se o tyto soubory:

- **aitken-koef-0st.R**
- **aitken-koef-1st.R**
- **aitken-koef-2st.R**
- **aitken-koef-3st.R**
- **aitken-koef-4st.R**
- **aitken-koef-5st.R**

Soubor **graf-interpol-poly.R** je v programu taktéž obsažen. Čtenář najde tento program na přiloženém CD.

Uživatel postupuje stejně jako Nevillova algoritmu. Zadává vstup do části *zadejte data*, kde jsou přichystané dva vektory $x_i$ a $f_i$. Zde vloží hodnoty a dle počtu zadaných dat vybere soubor pro výpočet koeficientů interpolačního polynomu. Pomocí příkazu *source* přikáže spustit tento soubor, poté pomocí příkazu *source* přikáže spustit soubor pro vykreslení grafu. *R* jako výstup vypíše vektor koeficientů a vykreslí graf.

Ukážeme si tento postup konkrétně na Příkladu 5.

Program **AITKEN-kalkulator.R** vypadá takto:

```
# ITEROVANA - AITKEN
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi <- c(-2,-1,0,1)
fi <- c(35,6,-3,-10)
### dle poctu vstupnich dat zvolte a spustte soubor pro hledani
### koeficientu:
```

```
### ( pro n vstupnich dat xi zvolte "aitken-koef-(n-1)st.R")
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("aitken-koef-0st.R")
source("aitken-koef-1st.R")
source("aitken-koef-2st.R")
source("aitken-koef-3st.R")
source("aitken-koef-4st.R")
source("aitken-koef-5st.R")
### spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

V Příkladu 5 máme čtyři vstupní data, budeme tedy hledat interpolační polynom stupně 3. Spustíme soubor **aitken-koef-3st.R** pro nalezení koeficientů interpolačního polynomu stupně 3. Dále spustíme soubor **graf-interpol-poly.R** pro vykreslení grafu, který nám vykreslil tentýž graf, jako vykreslil u Nevillova algoritmu. Výstupem je vektor koeficientů a obrázek s grafem, viz Obrázek 4 (analogický jako obrázek s grafem pro Příklad 4).

```
> source("aitken-koef-3st.R")
[1] -3  1 -5 -3
```

Na ukázku uvádím kód ze souboru **aitken-koef-3st.R**, který jsem použila ke hledání interpolačního polynomu z Příkladu 5. Obecný kód pro vykreslení grafu hledaného interpolačního polynomu je analogický jako u ostatních metod. Zbývající kódy jsou k dispozici na CD.

```
#ITEROVANA INTERPOLACE - AITKENUV ALGORITMUS
# vypocet koeficientu pro interpolacni polynom stupně 3
### overeni podminky:
n <- length(xi)
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
```

```
### pomocne promenne:
x <- seq(from=min(xi)-0.1*abs(min(xi)),
         to=max(xi)+0.1*abs(max(xi)),length=100)
### koeficienty postupne konstruovanych polynomu:
#P01
a <- c(1:2)
a[1] <-             (fi[2]-fi[1])/(xi[2]-xi[1])
a[2] <- (fi[1]*xi[2]-fi[2]*xi[1])/(xi[2]-xi[1])
#P02
b <- c(1:2)
b[1] <-             (fi[3]-fi[1])/(xi[3]-xi[1])
b[2] <- (fi[1]*xi[3]-fi[3]*xi[1])/(xi[3]-xi[1])
#P03
c <- c(1:2)
c[1] <-             (fi[4]-fi[1])/(xi[4]-xi[1])
c[2] <- (fi[1]*xi[4]-fi[4]*xi[1])/(xi[4]-xi[1])
#P012
f <- c(1:3)
f[1] <- (b[1]-a[1])                      /(xi[3]-xi[2])
f[2] <- (b[2]-a[2]-b[1]*xi[2]+a[1]*xi[3])/(xi[3]-xi[2])
f[3] <-          (-b[2]*xi[2]+a[2]*xi[3])/(xi[3]-xi[2])
#P013
g <- c(1:3)
g[1] <- (c[1]-a[1])                      /(xi[4]-xi[2])
g[2] <- (c[2]-a[2]-c[1]*xi[2]+a[1]*xi[4])/(xi[4]-xi[2])
g[3] <-          (-c[2]*xi[2]+a[2]*xi[4])/(xi[4]-xi[2])
#P0123
k <- c(1:4)
k[1] <- (g[1]-f[1])                      /(xi[4]-xi[3])
k[2] <- (g[2]-f[2]-g[1]*xi[3]+f[1]*xi[4])/(xi[4]-xi[3])
```

```
k[3] <- (g[3]-f[3]-g[2]*xi[3]+f[2]*xi[4])/(xi[4]-xi[3])
k[4] <-          (-g[3]*xi[3]+f[3]*xi[4])/(xi[4]-xi[3])
k <- print (k)
```

# 5. Hermitovská interpolace

V této kapitole budu studovat Hermitovskou interpolaci, pro kterou jsem čerpala z [2] a [4]. I zde uvedu vlastní příklad vyřešený jak ručně, tak i v $R$. Pro tento příklad taktéž vykreslím graf.

Tato metoda interpolace používá k výpočtu interpolačního polynomu navíc i hodnoty derivací. Nechť jsou dány uzly $x_0, \ldots, x_n$, pro které platí $x_i \neq x_k$ pro $i \neq k$, funkční hodnoty $f_i$ a hodnoty derivací $f_i'$, kde $i = 0, 1, \ldots, n$. O Hermitově interpolačním polynomu hovoříme v případě, že chceme najít polynom stupně $2n + 1$ tak, aby byly splněny podmínky:

$$p_{2n+1}(x_i) = f_i, \;\; p'_{2n+1}(x_i) = f_i' \;\; \forall i = 0, 1, \ldots, n. \tag{11}$$

Následující věta nám říká, jak takový polynom sestrojit.

**Věta 4** *Nechť jsou dány body $x_0, \ldots, x_n$, $x_i \neq x_k$ pro $i \neq k$, funkční hodnoty $f_i$ a hodnoty prvních derivací $f_i'$, $\forall i = 0, 1, \ldots, n$. Hermitovým interpolačním polynomem rozumíme polynom $p_{2n+1}(x)$ stupně $2n + 1$, který splňuje:*

$$p_{2n+1}(x_i) = f_i, \;\; p'_{2n+1}(x_i) = f_i' \;\; \forall i = 0, 1, \ldots, n.$$

*Lze jej vyjádřit ve tvaru:*

$$p_{2n+1}(x) = \sum_{i=0}^{n} h_i(x) f_i + \sum_{i=0}^{n} \overline{h_i}(x) f_i', \tag{12}$$

*kde*

$$h_i(x) = \left[1 - 2(x - x_i) l_i'(x_i)\right] l_i^2(x), \tag{13}$$

$$\overline{h_i}(x) = (x - x_i) l_i^2(x), \tag{14}$$

*přičemž $l_i(x)$ jsou Lagrangeovy fundamentální polynomy dané vztahem (6).*

**Důkaz 4** *: viz [2], str. 184.*

**Příklad 6** Sestrojte Hermitův interpolační polynom pro data:

| $i$ | 0 | 1 | 2 |
|---|---|---|---|
| $x_i$ | $-1$ | 0 | 1 |
| $f_i$ | $-8$ | $-1$ | 10 |
| $f_i'$ | 19 | 6 | 15 |

*Řešení:* hledáme polynom $p_5(x)$, jelikož $n = 2$.

Vypočítáme si fundamentální polynomy dle vztahu (6), jejich derivace a funkční hodnoty v těchto derivacích:

$$l_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-0)(x-1)}{(-1-0)(-1-1)} = \frac{1}{2}(x^2-x),$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-(-1))(x-1)}{(0-(-1))(0-1)} = 1-x^2,$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-(-1))(x-0)}{(1-(-1))(1-0)} = \frac{1}{2}(x^2+x).$$

Odtud plyne:

$$\begin{array}{ll} l_0'(x) = x - \frac{1}{2} & l_0'(x_0) = -\frac{3}{2} \\ l_1'(x) = -2x & l_1'(x_1) = 0 \\ l_2'(x) = x + \frac{1}{2} & l_2'(x_2) = \frac{3}{2}. \end{array}$$

Nyní dle vztahu (13) najdeme polynomy $h_i(x)$ pro $i = 0, 1, 2$.

$$\begin{aligned} h_0(x) &= [1 - 2(x-x_0)l_0'(x_0)]\, l_0^2(x) = \\ &= \left[1 - 2(x-(-1))\left(-\frac{3}{2}\right)\right]\left(\frac{1}{2}(x^2-x)\right)^2 = \\ &= \frac{1}{4}(3x^5 - 2x^4 - 5x^3 + 4x^2), \end{aligned}$$

$$\begin{aligned} h_1(x) &= [1 - 2(x-x_1)l_1'(x_1)]\, l_1^2(x) = \\ &= [1 - 2(x-0)(0)]\,(1-x^2)^2 = \\ &= x^4 - 2x^2 + 1, \end{aligned}$$

$$
\begin{aligned}
h_2(x) &= \left[1 - 2(x - x_2)l_2'(x_2)\right] l_2^2(x) = \\
&= \left[1 - 2(x-1)\frac{3}{2}\right] \left(\frac{1}{2}(x^2 + x)\right)^2 \\
&= \frac{1}{4}(-3x^5 - 2x^4 + 5x^3 + 4x^2).
\end{aligned}
$$

Podobně dle vztahu (14) najdeme i polynomy $\overline{h_i}(x)$, pro $i = 0, 1, 2$.

$$
\begin{aligned}
\overline{h_0}(x) &= (x - x_0)l_0^2(x) \\
&= (x - (-1)) \left(\frac{1}{2}(x^2 - x)\right)^2 \\
&= \frac{1}{4}(x^5 - x^4 - x^3 + x^2), \\
\overline{h_1}(x) &= (x - x_1)l_1^2(x) \\
&= (x - 0)(1 - x^2)^2 \\
&= x^5 - 2x^3 + x, \\
\overline{h_2}(x) &= (x - x_2)l_2^2(x) \\
&= (x - 1) \left(\frac{1}{2}(x^2 + x)\right)^2 \\
&= \frac{1}{4}(x^5 + x^4 - x^3 - x^2).
\end{aligned}
$$

Nakonec dosadíme do vzorce (12) a dostaneme hledaný interpolační polynom stupně 5, který je vykreslen na Obrázku 5.

$$
\begin{aligned}
p_5(x) &= f_0 h_0(x) + f_1 h_1(x) + f_2 h_2(x) + f_0{}'\overline{h_0}(x) + f_1{}'\overline{h_1}(x) + f_2{}'\overline{h_2}(x) = \\
&= (-8)\,\frac{1}{4}\,(x^5 - 2x^4 - 5x^3 + 4x^2) + (-1)\,(x^4 - 2x^2 + 1) + \\
&\quad + 10\,\frac{1}{4}\,(-3x^5 - 2x^4 + 5x^3 + 4x^2) + 19\,\frac{1}{4}(x^5 - x^4 - x^3 + x^2) + \\
&\quad + 6\,(x^5 - 2x^3 + x) + 15\,\frac{1}{4}\,(x^5 + x^4 - x^3 - x^2) = \\
&= x^5 - 3x^4 + 2x^3 + 5x^2 + 6x - 1.
\end{aligned}
$$

## 5.1. Hermitovská interpolace metodou neurčitých koeficientů

Interpolační polynom pro data z Příkladu 6 je možné najít i jinou cestou. Lze v tomto případě použít metodu neurčitých koeficientů. Tento postup hledání interpolačního polynomu pro stejná data si ukážeme v následujícím Příkladu 7.

Budeme hledat polynom stupně $2n+1$, který má splňovat interpolační podmínky (11). A protože jsou v tomto případě přítomny i funkční hodnoty prvních derivací, lze je rozepsat do následujících tvarů:

$$\begin{aligned} p_{2n+1}(x_0) &= a_0 x_0^{2n+1} + a_1 x_0^{2n} + \ldots + a_{2n} = f_0 \\ p_{2n+1}(x_1) &= a_0 x_1^{2n+1} + a_1 x_1^{2n} + \ldots + a_{2n} = f_1 \\ &\vdots \\ p_{2n+1}(x_n) &= a_0 x_n^{2n+1} + a_1 x_n^{2n} + \ldots + a_{2n} = f_n. \end{aligned}$$

$$\begin{aligned} p'_{2n+1}(x_0) &= (2n+1)\, a_0 x_0^{2n} + (2n) a_1 x_0^{2n-1} + \ldots + a_{2n-1} = f'_0 \\ p'_{2n+1}(x_1) &= (2n+1)\, a_0 x_1^{2n} + (2n) a_1 x_1^{2n-1} + \ldots + a_{2n-1} = f'_1 \\ &\vdots \\ p'_{2n+1}(x_n) &= (2n+1)\, a_0 x_n^{2n} + (2n) a_1 x_n^{2n-1} + \ldots + a_{2n-1} = f'_n. \end{aligned}$$

Dostáváme soustavu $(2n+2)$ lineárních rovnic o $(2n+2)$ neznámých ve tvaru:

$$a_0 x_i^{2n+1} + a_1 x_i^{2n} + \ldots + a_{2n} = f_i,\ i = 0, 1, \ldots, n, \tag{15}$$

$$(2n+1)\, a_0 x_i^{2n} + (2n) a_1 x_i^{2n-1} + \ldots + a_{2n-1} = f'_i,\ i = 0, 1, \ldots, n. \tag{16}$$

I takovou soustavu můžeme vyřešit např. již zmíněnou Gaussovou eliminační metodou.

**Příklad 7** Sestrojte Hermitův interpolační polynom pro data:

| $i$ | 0 | 1 | 2 |
|---|---|---|---|
| $x_i$ | $-1$ | 0 | 1 |
| $f_i$ | $-8$ | $-1$ | 10 |
| $f'_i$ | 19 | 6 | 15 |

*Řešení:* hledáme polynom $p_5(x)$, jelikož $n = 2$. Dle vztahu (15) a (16):

$$\begin{aligned}
a_0(-1)^5 + a_1(-1)^4 + a_2(-1)^3 + a_3(-1)^2 + a_4(-1)^1 + a_5 &= -8 \\
a_0(0)^5 + a_1(0)^4 + a_2(0)^3 + a_3(0)^2 + a_4(0)^1 + a_5 &= -1 \\
a_0(1)^5 + a_1(1)^4 + a_2(1)^3 + a_3(1)^2 + a_4(1)^1 + a_5 &= 10 \\
5a_0(-1)^4 + 4a_1(-1)^3 + 3a_2(-1)^2 + 2a_3(-1)^1 + 1a_4 &= 19 \\
5a_0(0)^4 + 4a_1(0)^3 + 3a_2(0)^2 + 2a_3(0)^1 + 1a_4 &= 6 \\
5a_0(1)^5 + 4a_1(1)^3 + 3a_2(1)^2 + 2a_3(1)^1 + 1a_4 &= 15
\end{aligned}$$

Řešení této soustavy provedeme např. Gaussovou eliminační metodou a dostaneme koeficienty $a_0$ až $a_5$. Tj. $a_0 = 1$, $a_1 = -3$, $a_2 = 2$, $a_3 = 5$, $a_4 = 6$, $a_5 = -1$. Interpolační polynom je tedy ve tvaru $p_5(x) = x^5 - 3x^4 + 2x^3 + 5x^2 + 6x - 1$ (viz Obrázek 5). Podmínky interpolace můžeme ověřit dosazením předepsaných bodů z tabulky do interpolačního polynomu.

$$\begin{aligned}
1(-1)^5 + (-3)(-1)^4 + 2(-1)^3 + 5(-1)^2 + 6(-1)^1 + (-1) &= -8 \\
1(0)^5 + (-3)(0)^4 + 2(0)^3 + 5(0)^2 + 6(0)^1 + (-1) &= -1 \\
1(1)^5 + (-3)(1)^4 + 2(1)^3 + 5(1)^2 + 6(1)^1 + (-1) &= 10 \\
5(1))(-1)^4 + 4(-3)(-1)^3 + 3(2)(-1)^2 + 2(5)(-1)^1 + 1(6) &= 19 \\
5(1)(0)^4 + 4(-3)(0)^3 + 3(2)(0)^2 + 2(5)(0)^1 + 1(6) &= 6 \\
5(1)(1)^5 + 4(-3)(1)^3 + 3(2)(1)^2 + 2(5)(1)^1 + 1(6) &= 15
\end{aligned}$$

Tímto jsme si ověřili, že jsou splněny interpolační podmínky. Řešení bylo tedy správné.

V *R* jsem pro nalezení interpolačního polynomu metodou Hermitovské interpolace vytvořila program **HERMIT-kalkulator.R** . Pro tvorbu kódu jsem zvolila jednodušší cestu, a to naprogramování prostřednictvím metody neurčitých koeficientů. Program obsahuje dva soubory:

- **hermit-koef.R**
- **graf-interpol-poly.R**

První soubor je obecný kód pro výpočet koeficientů interpolačního polynomu, seřazených od $a_0$ až po $a_{2n+1}$. Druhý soubor je kódem pro vykreslení grafu interpolačního polynomu. Čtenář tento program najde na přiloženém CD.

Uživatel postupuje podobně jako u metody neurčitých koeficientů. Zadává pouze vstup do části *zadejte data*. Hodnoty vloží do vektorů $x_i$, $f_i$ a $df_i$, které jsou zde přichystány. Hodnoty $df_i$ jsou funkční hodnoty derivací v bodech $x_i$. Pomocí příkazu *source* přikáže spustit oba soubory. *R* mu jako výstup vypíše vektor koeficientů a vykreslí graf.

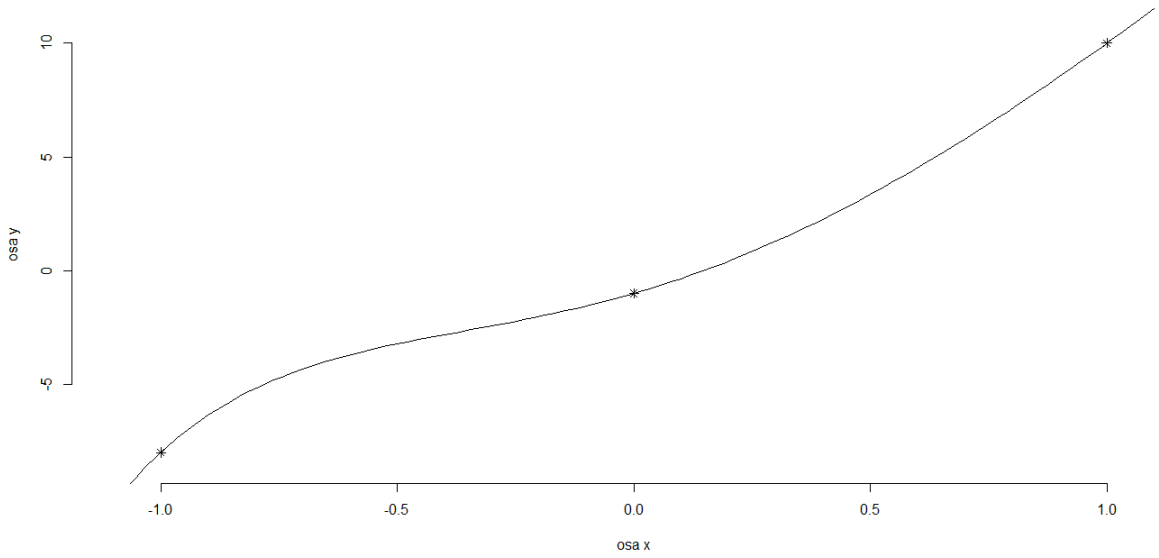
Ukážeme si tento postup konkrétně na Příkladu 7.

Program **HERMIT-kalkulator.R** vypadá takto:

```
#HERMIT POMOCI MNK
### nastaveni pracovniho adresare:
### nutnost nastaveni cesty do slozky Interpolacni soubory na CD
### např. setwd("E:/Interpolace v R/Interpolacni soubory")
setwd("")
### zadejte data:
### xi[i] musi byt ruzne od xi[k] pro kazde i ruzne od k !
xi  <- c(-1,0,1)
fi  <- c(-8,-1,10)
dfi <- c(19,6,15)
### spustte soubor pro hledani koeficientu:
### ( koeficienty jsou razeny dle vztahu:
###   p(n-1) = k[1]*x^(n-1) + k[2]*x^(n-2) + ... + k[n]*x^0 )
source("hermit-koef.R")
### pote spustte soubor pro vykresleni interpolacniho polynomu:
source("graf-interpol-poly.R")
```

V Příkladu 7 máme tři vstupní data, tj. $n = 2$, hledaným inerpolačním polynomem je proto interpolační polynom stupně 5. Spustíme soubor **hermit-koef.R** pro nalezení koeficientů interpolačního polynomu, a následně i soubor **graf-interpol-poly.R** pro vykreslení grafu. Jako výstup dostaneme vektor koeficientů a obrázek, viz Obrázek 5, na kterém je vykreslen graf.

```
> source("hermit-koef.R")
[1]  1 -3  2  5  6 -1
```




Obrázek 5: Graf interpolačního polynomu určeného Hermitovskou interpolací pomocí metody neurčitých koeficientů pro data z Příkladu 6 a 7

Opět uvádím náhled na obecný kód **hermit-koef.R**, který byl použitý ke hledání koeficientů interpolačního polynomu z Příkladu 7. Obecný kód pro vykreslení grafu hledaného interpolačního polynomu je opět stejný jako u metody neurčitých koeficientů.

**hermit-koef.R**:

```
# HERMITOVSKA INTERPOLACE POMOCI MNK
# vypocet koeficientu pro interpolacni polynom stupně n-1
```

```
### overeni podminky:
n <- length(xi)
m <- length(fi)
if(n!=m) stop("zadej stejne dlouhe vektory xi a fi!")
### pomocne promenne:
x <- seq(from=min(xi)-0.1*abs(min(xi)),
         to=max(xi)+0.1*abs(max(xi)),length=100)
l <- n-1
### definice matice K a vektoru gi:
K <- matrix(0, nrow = 2*l+2, ncol = 2*l+2)
for(j in 1:(2*l+2))
{K[1:n,j] <- xi^((2*l+2)-j)}
for(j in 1:(2*l+1))
{K[(n+1):(2*n),j] <- ((2*l+2)-j)*xi^((2*l+1)-j)}
gi <- c(fi,dfi)
### vypocet koeficientu:
p <- solve(K,gi)
print(k <- as.numeric(p))
```

# 6. Shrnutí

Hlavním záměrem této práce pro mě bylo ukázat čtenáři možnosti nalezení interpolačního polynomu, které zrekapituluji v této kapitole.

Jednou z možností je nalézt interpolační polynom ručním výpočtem. Vložila jsem zde sedm vlastních příkladů, které jsem podrobně vysvětlila tak, aby byl čtenář schopen porozumět způsobu hledání interpolačního polynomu konkrétní metodou.

V Příkladech 1, 2, 3, 4 a 5 je vysvětlena konstrukce interpolačního polynomu pro případ, kdy jsou zadány uzlové body $x_i$ a funkční hodnoty $f_i$. Tehdy lze použít metodu neurčitých koeficientů, Lagrangeovou metodou, Newtonovou metodou a metodou iterované interpolace, která obsahuje dva algoritmy. V případě, že jsou zadány uzlové body $x_i$, funkční hodnoty $f_i$, a navíc i funkční hodnoty prvních derivací $f'_i$ uvádím Příklad 6 pro hledání interpolačního polynomu Hermitovskou metodou. Takový interpolační polynom však lze nalézt i metodou neurčitých koeficientů, což je předvedeno v Příkladu 7.

Další možností je použít některý z programů, které jsem vytvořila v *R*. Je zde k dispozici šest kalkulátorů, které vždy najdou koeficienty interpolačního polynomu pro daná data a vykreslí pro takový interpolační polynom graf. Uživateli nabízím kalkulatory **MNK-kalkulator.R** a **HERMIT-kalkulator.R**, které najdou koeficienty interpolačního polynomu jakéhokoliv stupně. Dále nabízím kalkulátory **LAGRANGE-kalkulator.R**, **NEWTON-kalkulator.R**, **NEVILL-kalkulator.R** a **AITKEN-kalkulator.R**, které můžou najít koeficienty pro interpolační polynom stupně 0 až 5. Tyto kalkulátory navíc nabízí i koeficienty např. fundamentálních Lagrangeových polynomů nebo postupně konstruovaných

interpolačních polynomů v případě iterované interpolace.

Při práci s $R$ jsem také narazila na několik užitečných funkcí. Např. můžu doporučit funkci

```
coef,
```

která nám vypíše vektor koeficientů interpolačního polynomu pro daná data. Je nutné však dodržet vztah mezi počtem bodů a stupněm interpolačního polynomu, viz Věta 1. Funkce je naprogramována metodou nejmenších čtverců [4], [3]. Například pro data z Příkladu 1:

```
xi <- c(-1,0,2,5)
fi <- c(10,3,13,-2)
```

vypíše koeficienty tímto způsobem:

```
> k = coef(lm(fi ~ I(xi^1) + I(xi^2) + I(xi^3)))
> print(k)
(Intercept)      I(xi^1)      I(xi^2)      I(xi^3)
          3           -1            5           -1
```

Koeficienty řadí v opačném pořadí než používáme my, viz (1), a to od $a_n$ až po $a_0$.

Využití polynomů k interpolaci je výhodné, protože s polynomy se dobře pracuje. Jak už ale bylo řečeno, polynomiální interpolace při interpolaci polynomem stupně vyššího než 3 ztrácí na významu a je výhodnější použít interpolaci spajny. Uvádím zde tři vlastní příklady, které vyřeším v $R$ pomocí obecného kódu **MNK-kalkulator.R**. Následně vykreslím grafy nalezených interpolačních polynomů stupně 6, 10 a 15, na kterých jsou vidět jasné známky oscilace vždy mezi prvními a posledními dvěma uzlovými body.

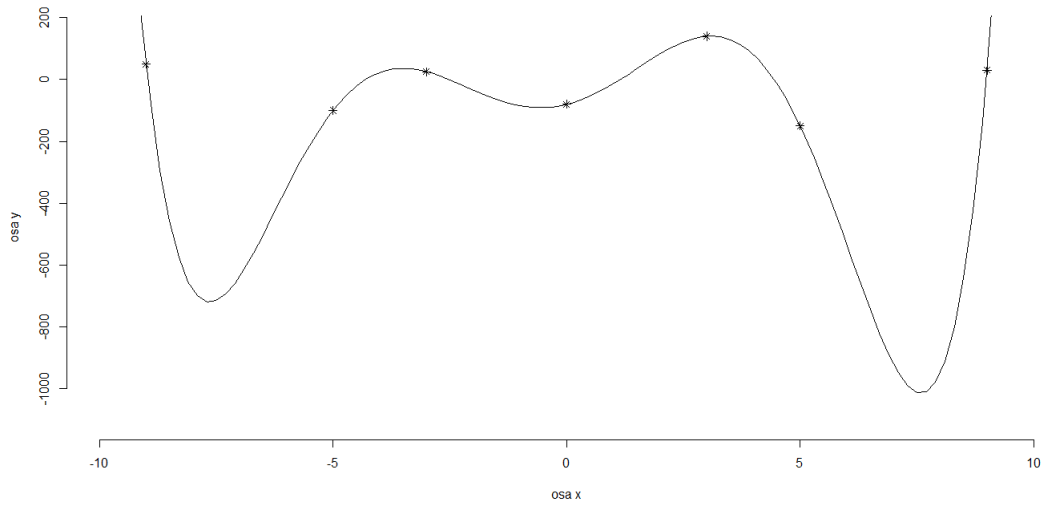
**Příklad 8** Najděte interpolační polynom pro daná data:

| $x_i$ | $-9$ | $-5$ | $-3$ | $0$ | $3$ | $5$ | $9$ |
|---|---|---|---|---|---|---|---|
| $f_i$ | $50$ | $-100$ | $25$ | $-80$ | $140$ | $-150$ | $30$ |

*Řešení:* hledáme polynom $p_6(x)$, jelikož $n = 6$.

Pro řešení jsem spustila oba soubory z **MNK-kalkulator.R**. Výstupem je vektor koeficientů a graf interpolačního polynomu $p_6(x)$ na Obrázku 6.

```
> source("mnk-koef.R")
[1]    0.01804958    0.02194252   -1.85465811   -2.25646219
[5]   33.28546214   37.69748264 -80.00000000
```




Obrázek 6: Graf interpolačního polynomu z Příkladu 8

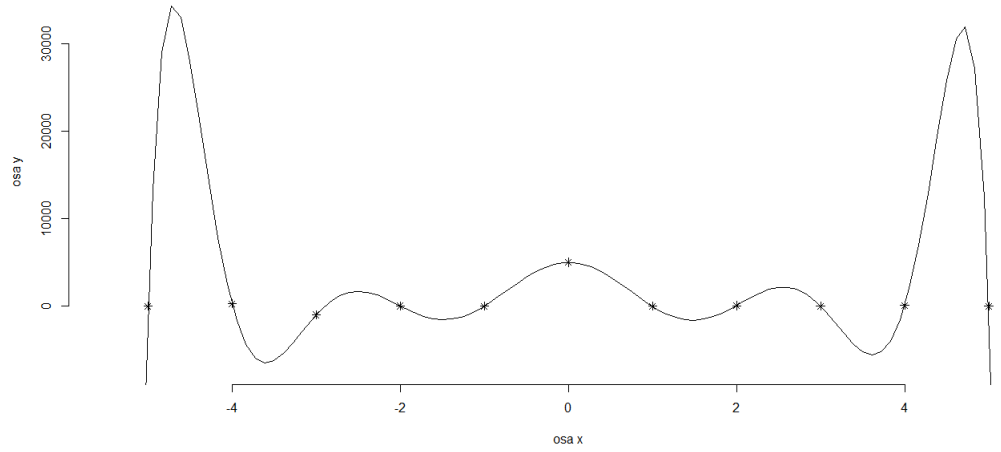
**Příklad 9** Najděte interpolační polynom pro daná data:

| $x_i$ | $-5$ | $-4$ | $-3$ | $-2$ | $-1$ | $0$ | $1$ | $2$ | $3$ | $4$ | $5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_i$ | $-4$ | $250$ | $-1000$ | $2$ | $-3$ | $5000$ | $-21$ | $52$ | $-3$ | $74$ | $0$ |

*Řešení:* hledáme polynom $p_{10}(x)$, jelikož $n = 10$.

Pro řešení jsem spustila oba soubory z **MNK-kalkulator.R**. Výstupem je vektor koeficientů a graf interpolačního polynomu $p_{10}(x)$ na Obrázku 7.

```
> source("mnk-koef.R")
 [1] -3.637288e-01  3.468778e-02  1.987030e+01 -1.557102e+00
 [5] -3.659480e+02  1.964615e+01  2.699631e+03 -6.131342e+01
 [9] -7.365189e+03  3.418968e+01  5.000000e+03
```




Obrázek 7: Graf interpolačního polynomu z Příkladu 9

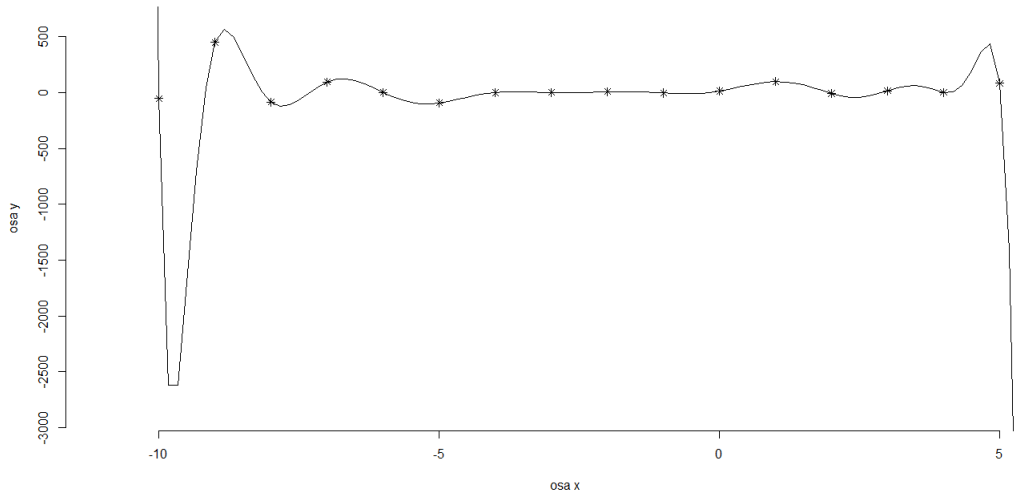
**Příklad 10** Najděte interpolační polynom pro data daná tabulkou:

| $x_i$ | $-10$ | $-9$ | $-8$ | $-7$ | $-6$ | $-5$ | $-4$ | $-3$ | $-2$ | $-1$ | $0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_i$ | $-52$ | $450$ | $-85$ | $96$ | $2$ | $-91$ | $0$ | $1$ | $8$ | $-3$ | $14$ |
| $x_i$ | $1$ | $2$ | $3$ | $4$ | $5$ | | | | | | |
| $f_i$ | $98$ | $-7$ | $17$ | $3$ | $84$ | | | | | | |

*Řešení:* hledáme polynom $p_{15}(x)$, jelikož $n = 15$.

Pro řešení jsem spustila oba soubory z **MNK-kalkulator.R**. Výstupem je vektor koeficientů a graf interpolačního polynomu $p_{15}(x)$ na Obrázku 8.

```
> source("mnk-koef.R")
 [1] -1.375870e-07 -4.594447e-06 -4.624234e-05  3.804371e-05
 [5]  3.233863e-03  1.181109e-02 -6.918480e-02 -4.360671e-01
 [9]  4.335613e-01  5.981876e+00  2.585914e+00 -3.405428e+01
[13] -3.527586e+01  6.199663e+01  8.282238e+01  1.400000e+01
```




Obrázek 8: Graf interpolačního polynomu z Příkladu 10

# Závěr

V této práci jsem čtenáři představila metody, kterými je možné nalézt interpolační polynom. Dále jsem uvedla vlastní vyřešené příklady pro konkrétní metody, které jsem doplnila o vysvětlení. Součástí práce jsou dávkové soubory, vztahující si k jednotlivým metodám. Pomocí těchto souborů můžeme nalézt koeficienty interpolačního polynomu pro konkrétní data a vykreslit si pro takový interpolační polynom graf. V závěrečné kapitole jsem shrnula možnosti nalezení interpolačního polynomu, které nabízí moje práce a upozornila jsem na své poznatky.

Tato práce pro mě byla velkým přínosem. Nejenže jsem si velmi prohloubila problematiku interpolace, ale také jsem se naučila pracovat s matematickým softwarem *R*. Navíc jsem dokázala získané znalosti aplikovat pro vytvoření něčeho, co může člověku výrazně ulehčit práci. Uživatelsky přívětivější bych však pro tento typ úloh označila matematický software *Matlab*. Zobecnění složitějších kódů z mojí práce by mohlo být námětem pro další práci, stejně jako vytvoření kódů pro interpolaci spajny.

Vytvořené dávkové soubory jsou k dispozici na přiloženém CD. Bakalářská práce byla vysázena systémem LATEX.

# Literatura


[1] Dvořáková Ľ., : *Lineární algebra 1* České vysoké učení technické v Praze, Praha, 2014

[2] Horová I., Zelinka J.:*Numerické metody* Masarykova univerzita, Brno, 2004

[3] Machalová J.:*Základy numerických metod* Univerzita Palackého, Olomouc, 2014 - e-learning

[4] Stoer J., Bulirsch R.: *Introduction to numerical Analysis*. Springer 2002

[5] The R Project for Statistical Computing [online], dostupné z WWW: http://www.rproject.org/ [cit. duben 2017]

[6] Scott, Theresa A.: An Introduction to R [online] 2004.[cit. duben 2017], dostupné z WWW: http://www.karlin.mff.cuni.cz/ kulich/vyuka/Rdoc/

[7] Bína, V., Komárek, A., Komárková, L.: Jak na jazyk R [online] 2006. [cit. duben 2017], dostupné z WWW: http://www.karlin.mff.cuni.cz/ komarek/Rko/Rmanual2.pdf RLectureTScott.pdf