Základní funkce ekonomické analýzy v prostředí Mathematica

Milan Cibulka

Bakalářská práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Milan Cibulka**
Osobní číslo:	**A13145**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Základní funkce ekonomické analýzy v prostředí Mathematica.**
Téma anglicky:	**The Basic Functions of Economic Analysis in the Mathematica Environment**

Zásady pro vypracování:

1. **Uveďte krátký přehled relevantních ekonomických funkcí.**
2. **Objasněte vzájemnou souvislost některých ekonomických funkcí na modelových příkladech.**
3. **Klasifikujte vybrané ekonomické funkce z hlediska matematické analýzy.**
4. **Napište program v prostředí Mathematica, který umožní vizualizaci zvolených ekonomických funkcí.**
5. **Napište program v prostředí Mathematica, který bude řešit modelové příklady na minimalizaci nákladů, maximalizaci zisku a vztahy mezi poptávkou a nabídkou.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **CHRAMCOV, Bronislav. Základy práce v prostředí Mathematica. Zlín: UTB FAI, 2006. 122 s. ISBN 80-7318-510-5**
2. **WOLFRAM, Stephen. The mathematica book. 5th ed. Champaign, IL : Wolfram Media, 2003. 1464 s. ISBN 1579550223.**
3. **Wolfram Language andSystem documentation center [online]. 2017 [cit. 2017-02-03]. Dostupné z WWW: [http://reference.wolfram.com/language/].**
4. **ALLEN, R.G.D. Matematická ekonomie. Praha: ACADEMIA, 1971. 784 s. ISBN nemá.**
5. **SEKERKA, Bohuslav. Mikroekonomie matematické a kvantitativní základy. Profess Consulting, 2002. 361 s. ISBN 80-7259-030-8**

Vedoucí bakalářské práce: **RNDr. Martin Fajkus, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **24. února 2017**

Termín odevzdání bakalářské práce: **24. května 2017**

Ve Zlíně dne 24. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Jméno, příjmení: Milan Cibulka

Název bakalářské/diplomové práce: Základní funkce ekonomické analýzy v prostředí Mathematica

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářsképráci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledkůbudu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 23. 5. 2017

Cibulka
.....
podpis diplomanta

ABSTRAKT

Tato bakalářská práce se věnuje vývoji interaktivní aplikace v prostředí Wolfram Mathematica, umožňující snadnou vizualizaci ekonomických funkcí a vztahů mezi nimi. Práce obsahuje stručný přehled a vysvětlení základních ekonomických funkcí. Zmíněny jsou také vybrané příkazy z prostředí Mathematica.

Klíčová slova: Wolfram Mathematica, náklady, příjmy, ekonomická funkce, minimalizace, maximalizace

ABSTRACT

This bachelor's thesis deals with the development of an interactive application in the environment of the program Wolfram Mathematica, which enables easy visualization of economic functions and their relations. The thesis includes concise overview and explanation of basic economical functions. There are also mentioned selected commands from the program Mathematica.

Keywords: Wolfram Mathematica, costs, revenues, Economic function, minimization, maximization

Tímto bych rád poděkoval panu RNDr. Martinu Fajkusovi, Ph.D. za odborné vedení práce a vstřícnost při konzultacích.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tato bakalářská práce se zabývá matematickou analýzou ekonomických funkcí a vztahů mezi nimi. Hlavním předmětem je vytvoření aplikace pro snadnou vizualizaci funkcí a řešení úloh na minimalizaci nákladů, maximalizaci zisku a řešení vztahů mezi poptávkou a nabídkou. Práce je rozdělena na teoretickou a praktickou část.

Teoretická část uvádí přehled základních ekonomických funkcí a stručně analyzuje jejich matematický význam. Teoretickou část doplňuje rozbor vybraných příkazů z prostředí Wolfram mathematica.

Praktickou část tvoří popis kódu funkcí vytvořených pro potřeby zhotovené aplikace. Je zde také podrobný popis uživatelského rozhraní a možností na nastavení. Funkčnost aplikace je prezentována na konkrétním příkladu.

Aplikace je vhodná i pro uživatele bez znalosti práce v prostředí Wolfram mathematica, veškeré nastavení je řešeno jednoduchými ovládacími prvky v uživatelském rozhraní aplikace.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ FUNKCE EKONOMICKÉ ANALÝZY

V ekonomické praxi se velmi často řeší optimalizační problémy např. maximalizace celkových příjmů, nebo minimalizace celkových nákladů. Při řešení optimalizačních problémů jde tedy o zjištění hodnot nezávisle proměnné, při nichž daný tržní subjekt maximalizuje či minimalizuje svoji cílovou funkci. Matematicky jde o hledání extrémů funkce. [66666]

1.1 Aplikace diferenciálního počtu v ekonomii

1.1.1 Funkce celkových a průměrných nákladů

Funkce celkových nákladů $TC(x)$ na vyrobení x výrobků je dána součtem fixních dů FC a variabilních nákladů $VC(x)$. Fixní náklady jsou konstantní a jejich výše se nemění s výší výroby (např. nájemné), kdežto variabilní náklady jsou závislé na výši výroby (čím vyšší výroba tím větší náklady, funkce má rostoucí průběh).

$$TC(x) = FC + VC(x)$$

Funkci průměrných nákladů $AC(x)$ získáme jednoduše podělením funkce celkových nákladů $TC(x)$ počtem výrobků x.[7]

$$AC(x) = \frac{TC(x)}{x}$$

1.1.1.1 Minimalizace průměrných nákladů

Nalezením minima funkce průměrných nákladů $AC(x)$ lze získat počet výrobků x_0 který zabezpečuje nejnižší náklady na výrobu.

Například pokud je dána funkce celkových nákladů $TC(x)$ ve tvaru:

$$TC(x) = 0{,}5x^2 + 98$$

Pro funkci průměrných nákladů platí:

$$AC(x) = \frac{TC(x)}{x} = \frac{0{,}5x^2 + 98}{x} = 0{,}5x + \frac{98}{x}$$

Pro nalezení minima funkce je nutno funkci průměrných nákladů derivovat a výslednou derivaci položit rovnu nule.

$$0{,}5 - \frac{98}{x^2} = 0$$

Hledání v intervalu (0,∞) odpovídá kořen rovnice:

$$x_0 = 14$$

Pokud se vezme v úvahu, že bod x_0 je stacionárním bodem, $AC'(x)$ existuje na celém intervalu (0,∞), druhá derivace v bodě x_0 je větší než nula $AC''(14) > 0$ a celkový charakter funkce, tak lze tvrdit, že v bodě x_0 je globální minimum na intervalu (0,∞). Z čehož vyplývá, že při výrobě čtrnácti kusů budou zajištěny nejnižší průměrné náklady.[7]

1.1.2 Funkce celkových a průměrných příjmů

Výpočtem funkce celkových příjmů $TR(x)$ je množství peněz, které výrobce získá prodejem výrobků. Vztah je dán součinem ceny výrobku p a počtu prodaných výrobků x.

$$TR(x) = px$$

Funkce průměrných příjmů $AR(x)$ je definována jako podíl funkce celkových příjmů $TR(x)$ a počtu prodaných výrobků x. [5]

$$AR(x) = \frac{TR(x)}{x}$$

1.1.2.1 Maximalizace celkových příjmů

Nalezením maxima funkce $TR(x)$ lze získat počet výrobků x, který zaručuje maximální příjem.

Například pokud je dána funkce celkových poptávky ve tvaru:

$$p = 10 - x$$

Pro funkci celkových příjmů platí:

$$TR(x) = px = -x^2 + 10x$$

Pro nalezení maxima funkce je nutno položit první derivaci funkce rovnu nule.

$$TR'(x) = -2x + 10$$

$$-2x + 10 = 0$$

Řešením rovnice je $x_0 = 5$. Pokud se vezme v úvahu, že bod x_0 je stacionárním bodem, druhá derivace v bodě x_0 je menší než nula $TR''(5) < 0$ a fakt že jde o kvadratickou funk-

ci, lze tvrdit, že v bodě x_0 je globální maximum funkce. Z čehož vyplývá, že výrobce dosáhne maximálního příjmu v případě, že vyrobí 5 kusů výrobku. [6]

1.1.3 Funkce poptávky

Funkce poptávky vyjadřuje vztah mezi množstvím výrobků x, které jsou spotřebitelé ochotni koupit a cenou p daného výrobku. Jelikož spotřebitelé chtějí kupovat levné výrobky, tak pro poptávku platí pravidlo, že čím nižší bude cena výrobku tím více spotřebitel si výrobek koupí. Z toho vyplívá, že funkce poptávky bude mít vždy klesající průběh.

Může být reprezentována vztahem vyjadřujícím množství výrobků x v závislosti na ceně výrobku p.[5]

$$x = D(p)$$

Nebo inverzním vztahem vyjadřujícím cenu výrobku p v závislosti na množství výrobků x.

$$p = d(x)$$

1.1.4 Funkce nabídky

Funkce nabídky vyjadřuje vztah mezi cenou výrobku p a množstvím výrobků x, které je výrobce připraven vyrobit. Zájmem výrobce je prodat své co nejvíce výrobků za co nejvyšší cenu. Z toho vyplívá, že funkce nabídky bude mít vždy rostouc průběh.

Může být vyjádřena vztahem závislosti množství výrobků x na ceně výrobku p.[7]

$$x = S(p)$$

Nebo inverzním vztahem vyjadřujícím závislost ceny výrobku p na množství výrobků x.

$$p = s(x)$$

1.1.5 Rovnováha trhu

Funkce poptávky a nabídky spolu úzce souvisí a vytvářejí rovnováhu trhu, což označuje stav kdy množství výrobků žádané spotřebiteli je rovno množství výrobků nabízeného na trhu. Pokud jsou funkce nabídky a poptávky zadané vztahy:

$$x = S(p)$$

$$x = D(p)$$

Je možno postavit funkce do rovnosti:

$$D(p) = S(p)$$

a řešením výsledné rovnice získat rovnovážnou cenu výrobku a dosazením do jednoho ze vztahů získat i rovnovážné množství (souřadnice průsečíku funkcí).[4]

1.1.6 Marginální (mezní) náklady a příjmy

Funkce marginálních nákladů $MC(x)$ v bodě $x-1$ přibližně určuje mezní hodnotu nákladů, které je nutno vynaložit na výrobu $x-t\acute{e}ho$ výrobku a je dána derivací funkce celkových nákladů.

$$MC(x) = TC'(x)$$

Přesné náklady $C(x)$ na výrobu $x-t\acute{e}ho$ výrobku se získají jako rozdíl celkových nákladů na výrobu x výrobků $TC(x)$ a nákladů na výrobu $x-1$ výrobků $TC(x-1)$.

$$C(x) = TC(x) - TC(x-1)$$

Což lze přibližně nahradit funkcí marginálních nákladů $MC(x)$ v bodě $x-1$.

$$C(x) \approx MC(x-1)$$

Pro marginální příjmy platí podobné vztahy jako pro marginální náklady. Funkce marginálních příjmů $MR(x)$ je definována jako derivace celkových příjmů $TR(x)$.

$$MR(x) = TR'(x)$$

Marginální příjmy $MR(x)$ v bodě $x-1$ přibližně určují mezní hodnotu příjmů které je možno získat z prodeje $x-t\acute{e}ho$ výrobku. Skutečný příjem $R(x)$ z prodeje $x-t\acute{e}ho$ výrobku je dán vztahem.

$$R(x) = TR(x) - TR(x-1)$$

Což pro zjednodušení lze nahradit funkcí marginálních příjmů $MR(x)$ v bodě $x-1$.[7]

$$R(x) \approx MR(x-1)$$

1.2 Aplikace integrálního počtu v ekonomii

1.2.1 Výpočet funkce celkových nákladů

Pokud je dána funkce marginálních nákladů $MC(x)$, lze z ní získat za pomoci neurčitého integrálu funkci celkových nákladů $TC(x)$, což už napovídá vztah $MC(x) = TC'(x)$, takže musí platit také tento vztah:

$$TC(x) = \int MC(x)\,dx$$

Kde po integraci je nutno nalézt hodnotu integrační konstanty C, která je rovna fixním nákladům FC. V případě že fixní náklady FC nejsou známy je možno je získat z hodnoty celkových nákladů na určité množství výrobků x. [6]

1.2.2 Výpočet funkce celkových příjmů

Funkci celkových příjmů $TR(x)$ je možné najít za pomoci funkce marginálních příjmů $MR(x)$ obdobně jako v předchozí kapitole. Pro marginální náklady $MR(x)$ platí vztah $MR(x) = TR'(x)$ a při použití neurčitého integrálu je možno vyjádřit funkci celkových příjmů $TR(x)$ jako:

$$TR(x) = \int MR(x)\,dx$$

Taktéž je po integraci nutno nalézt hodnotu integrační konstanty C, ta se určuje dosazením hodnoty 0 za x do výsledného předpisu funkce celkových příjmů $TR(x)$. Vychází se ze skutečnosti, že při prodeji nulového množství výrobků, vznikají nulové příjmy. [6]

1.2.3 Celkový příjem za zkoumané období

Funkce celkových příjmů $TC(x)$ určuje celkový příjem z prodeje x výrobků. Výpočet celkového příjmu za konkrétní období, pokud je dána hustota toku příjmu $f(t)$ v libovolném čase t zkoumaného období, se provede za pomoci určitého integrálu.

$$TR(T_1, T_2) = \int_{T_1}^{T_2} f(t)\,dt$$

Kde T_1 a T_2 jsou krajní hodnoty zkoumaného časového období (nejčastěji v řádu let).[6]

2 SOFTWARE WOLFRAM MATHEMATICA

Software *Wolfram Mathematica* je prostředí které spojuje prostředky pro řešení algebraických, numerických, grafických a jiných specifických úloh. Programovací jazyk *Mathematica* umožňuje velmi snadné vytvářen programů, jelikož není nutno deklarovat datové typy proměnných, jeden výraz může reprezentovat data, matematickou funkci, graf, zvuk nebo celý program. Velkou výhodou programového prostředí je také přítomnost interaktivní nápovědy, která nejenže obsahuje kompletní dokumentaci všech funkcí jazyka, ale přímo obsahuje příklady které je možno libovolně editovat a spouštět přímo v prostředí nápovědy [1, 2, 3].

2.1 Přehled některých funkcí jazyka *Mathematica*

Přehled funkcí a výrazů použitých v praktické části.

2.1.1 Práce s vektory

- *AppendTo[]*
  - Přidá na konec vektoru zvolený prvek (změní původní vektor).
- *Prepend[]*
  - Přidá na začátek vektoru zvolený prvek (nemění původní vektor).
- *Flatten[]*
  - Z tabulky vytvoří jednoduchý vektor.
- *Grid[]*
  - Vytváří z matice grafickou tabulku.
- *SpanFromLeft*
  - Výraz pro sloučení buňky tabulky s vedlejší buňkou vlevo.
- *Take[]*
  - Vrací submatici zvoleného rozměru.
- *Row[]*
  - Vrací vložené výrazy ve tvaru jednoho řádku.
- *MemberQ[]*
  - Podmínka kontrolující zda vektor obsahuje zadaný prvek.

2.1.2 Matematické funkce

- *Exp[]*
  - Matematická funkce e^x.
- *Log[]*
  - Matematická funkce $\ln x$.

2.1.3 Vytváření funkcí

- *Function[]*
  - Návratovou hodnotou této funkce je ryzí funkce jedné či více proměnných, vytvořená z výrazu na vstupu dle zadaných parametrů.
- *Module[]*
  - Důležitá funkce při vytváření vlastních funkcí. Umožňuje jednoduše vytvářet lokální proměnné v těle funkce a přiřazovat jim inicializační hodnoty.
- *Return[]*
  - Tato funkce ukončuje běh funkce a vrací hodnotu parametru z funkce, v niž je umístněna, bez parametru vrací *Null.*

2.1.4 Práce s textem

- *Style[]*
  - Funkce pro formátování textu, ale i grafických objektů, umožňuje širokou škálu nastavení. Lze nastavit barvu, velikost, tučné písmo, kurzívu a další možnosti formátování.
- *StringReplace[]*
  - Funkce, která v textu nahrazuje znak nebo řetězec, jiným znakem nebo řetězcem.
- *ToString[]*
  - Funkce pro převod výrazu na řetězec.
- *ToExpression[]*
  - Funkce pro převod řetězce na výraz.

2.1.5 Práce s grafy

- *Plot[]*
  - Funkce pro vykreslení grafu funkce jedné proměnné na určitém intervalu, nabízí velké množství grafického formátování.
- *ListPlot[]*
  - Funkce pro zobrazení bodů v grafu, možnosti formátování jsou obdobné jako u funkce *Plot[]*.
- *Show[]*
  - Funkce pro zobrazení kombinace grafických objektů, například umožňuje vykreslení více grafů do jednoho a lze i kombinovat různé druhy grafů,

2.1.6 Ovládací prvky

- *Slider[]*
  - Posuvník, nastavuje hodnotu výstupní proměnné na hodnotu v zadaném intervalu dle polohy posuvníku.
- *Checkbox[]*
  - Zaškrtávací pole, standardně nastavuje hodnotu výstupní proměnné na jednu ze dvou požadovaných hodnot, ale lze nastavit i více stavů než jen zaškrtnuto/nezaškrtnuto.
- *CheckboxBar[]*
  - Vytváří blok zaškrtávacích polí.
- *InputField[]*
  - Klasické vstupní pole, do výstupní proměnné se zapisuje obsah pole. Je možno definovat typ vstupu, jako řetězec nebo číslo. Pokud je zvolen typ vstupu číslo není možné přijímat žádné jiné znaky kromě čísel.
- *SetterBar[]*
  - Blok přepínacích tlačítek, hodnota výstupní proměnné se mění dle zvoleného tlačítka.
- *Button[]*
  - Standardní tlačítko, po kliknutí spouští předdefinovanou akci.

2.1.7 Uživatelské rozhraní

- *Manipulate[]*
  - Vytváří panel uživatelského rozhraní aplikace s ovládacími prvky a dynamicky zobrazuje výstup řízeného programu.
- *Delimiter*
  - Grafické oddělení objektů, použitelné například ve funkci *Manipulate[]* pro oddělení ovládacích prvků.
- *Panel[]*
  - Slouží jako podklad pro zobrazení libovolného objektu.
- *Dynamic[]*
  - Funkce zajišťující dynamický update proměnné.

2.1.8 Ostatní

- *Switch[]*
  - Funkce spouští část kódu určenou hodnotou řídíc proměnné.
- *If[]*
  - Funkce vyhodnotí podmínku a spouští kód pro splněnou podmínku nebo pro nesplněnou.
- *FindMinimum[]*
  - Funkce pro hledání minima matematické funkce.
- *FindMaximum[]*
  - Funkce pro hledání maxima matematické funkce.
- *FindFit[]*
  - Hledá koeficienty zadané rovnice funkce, aby výsledná funkce co nejlépe proložila zadané data.
- *Nsolve[]*
  - Funkce pro numerické řešení rovnic.

II. PRAKTICKÁ ČÁST

3 PROGRAM PRO VIZUALIZACI FUNKCÍ JEDNÉ PROMĚNNÉ

Základ programu tvoří funkce *Manipulate[]*, která zajišťuje grafické uživatelské prostředí s ovládacími prvky a zobrazení grafu požadovaných funkcí a výpočtů. Druhou nezbytnou součástí je funkce *program[]*, která je jádrem programu, a zajišťuje vizualizaci. Dále je zdrojový kód dělen do několika dílčích funkcí, které jsou využity především v těle funkce *program[]*.

3.1 Popis vytvořených funkcí

3.1.1 Funkce vytvářející matematické funkce

Funkce *createLinFunction[]* přebírá parametry a a b a vrací lineární funkci ve tvaru $y = a + bx$. Funkce *createPolyFunction2[]* přebírá parametry a, b a c a vrací polynomickou funkci druhého stupně ve tvaru $y = a + bx + cx^2$. Funkce *createPolyFunction3[]* přebírá parametry a, b, c a d a vrací polynomickou funkci třetího stupně ve tvaru $y = a + bx + cx^2 + dx^3$. Funkce *createPolyFunction4[]* přebírá parametry a, b, c, d a e a vrací polynomickou funkci čtvrtého stupně ve tvaru $y = a + bx + cx^2 + dx^3 + ex^4$. Funkce *createPolyFunction5[]* přebírá parametry a, b, c, d, e a f a vrací polynomickou funkci pátého stupně ve tvaru $y = a + bx + cx^2 + dx^3 + ex^4 + fx^5$.

```
createLinFunction[a_, b_] := Function[{x}, a + b * x]
createPolyFunction2[a_, b_, c_] := Function[{x}, a + b * x + c * x^2]
createPolyFunction3[a_, b_, c_, d_] := Function[{x}, a + b * x + c * x^2 + d * x^3]
createPolyFunction4[a_, b_, c_, d_, e_] := Function[{x}, a + b * x + c * x^2 + d * x^3 + e * x^4]
createPolyFunction5[a_, b_, c_, d_, e_, f_] := Function[{x}, a + b * x + c * x^2 + d * x^3 + e * x^4 + f * x^5]
createExpFunction[a_, b_] := Function[{x}, a + b * Exp[x]]
createLogFunction[a_, b_] := Function[{x}, a + b * Log[x]]
```

Obr. 1. Funkce pro tvorbu matematických funkcí

3.1.2 Funkce *getFitFunction[]*

Funkce *getFitFunction[]* dle parametrů typ, $data$ a $stupenPolynomu$ proloží data funkcí žádaného druhu a vrátí její rovnici. Parametr typ rozhoduje, zda bude výsledná funkce lineární, polynomická, exponenciální nebo logaritmická. Při volbě polynomické funkce rozhoduje parametr $stupenPolynomu$ jakého stupně bude fitovaná funkce, u ostatních typů se nijak nevyužívá. Potom co funkce *Switch[]* spustí příslušný blok kódu dle zvolené-

ho typu a případně stupně polynomu, tak se s pomocí funkce *FindFit[]* vyhledají nejvhodnější koeficienty žádané funkce, které se následně upraví na jednoduchý vektor. Nalezené koeficienty se vloží jako parametry do patřičné funkce popsané v předchozím odstavci a funkce *getFitFunction[]* vrátí výsledek.

```
getFitFunction[typ_, data_, stupenPolynomu_] := Module[{koeficienty = 0},
  Switch[typ,
    1, koeficienty = FindFit[data, a + b * x , {a, b}, x];
        koeficienty = koeficienty[[All, 2]];
        Return[createLinFunction[koeficienty[[1]], koeficienty[[2]]]];,
    2, Switch[stupenPolynomu,
          2, koeficienty = FindFit[data, a + b *x + c* x^2 , {a, b, c}, x];
              koeficienty = koeficienty[[All, 2]];
              Return[createPolyFunction2[koeficienty[[1]], koeficienty[[2]], koeficienty[[3]]]];,
          3, koeficienty = FindFit[data, a + b*x + c*x^2 + d*x^3, {a, b, c, d}, x];
              koeficienty = koeficienty[[All, 2]];
              Return[createPolyFunction3[koeficienty[[1]], koeficienty[[2]], koeficienty[[3]],
        koeficienty[[4]]]];,
          4, koeficienty = FindFit[data, a + b*x + c*x^2 + d*x^3 + e*x^4, {a, b, c, d, e}, x];
              koeficienty = koeficienty[[All, 2]];
              Return[createPolyFunction4[koeficienty[[1]], koeficienty[[2]], koeficienty[[3]],
        koeficienty[[4]], koeficienty[[5]]]];,
          5, koeficienty = FindFit[data, a + b *x + c* x^2 + d*x^3 + e*x^4 + f*x^5, {a, b, c, d, e, f}, x];
              koeficienty = koeficienty[[All, 2]];
              Return[createPolyFunction5[koeficienty[[1]], koeficienty[[2]], koeficienty[[3]],
        koeficienty[[4]], koeficienty[[5]], koeficienty[[6]]]];
          ];,
    3, koeficienty = FindFit[data, a + b * Exp[x] , {a, b}, x];
        koeficienty = koeficienty[[All, 2]];
        Return[createExpFunction[koeficienty[[1]], koeficienty[[2]]]];,
    4, koeficienty = FindFit[data, a + b * Log[x] , {a, b}, x];
        koeficienty = koeficienty[[All, 2]];
        Return[createLogFunction[koeficienty[[1]], koeficienty[[2]]]];
  ];
]
```

Obr. 2. Funkce getFitFunction[]

3.1.3 Funkce *getIntersections[]*

Funkce *getIntersections[]* slouží k vyhledání průsečíků dvou funkcí zadaných parametry $fce1$ a $fce2$. Funkce *NSolve[]* vyřeší kořeny rovnice $fce1 = fce2$, čímž získá $x-ové$ souřadnice hledaných průsečíků, které se následně upraví do vektoru. V cyklu *For[]* se dopočítají $y-ové$ souřadnice a vytvoří se matice souřadnic, která je konečným výstupem funkce *getIntersections[]*.

```
getIntersections[fce1_, fce2_] := Module[{pruseciky = 0, souradnice = {}, i},
  pruseciky = NSolve[fce1[x] == fce2[x], x, Reals];
  pruseciky = Flatten[pruseciky][[All, 2]];
  For[i = 1, i ≤ Length[pruseciky], i++,
   AppendTo[souradnice, {pruseciky[[i]], fce1[pruseciky[[i]]]}];
  ];
  Return[souradnice];
 ]
```

Obr. 3. Funkce getIntersection[]

3.1.4 Funkce *getTableOfIntersections*

Účel funkce *getTableOfIntersectios[]* je pouze vizualizační, parametry $prusecikyFit12$, $prusecikyFit1Y$, $prusecikyFit2Y$ přebírají matice průsečíků příslušných funkcí. Parametr $zobrazeni$ přebírá řídící vektor pro zobrazování objektů v grafickém uživatelském rozhraní. Při spuštění se nejprve vytvoří prázdná matice pouze s nadpisem a patřičným počtem sloupců dle počtu požadovaných průsečíků. Následně se dle čísel v řídícím vektoru přidají do počáteční matice tabulky se souřadnicemi průsečíků zvolených funkcí. Pokud vektor $zobrazeni$ obsahuje číslo 6, přidá se do matice tabulka s průsečíky mezi funkcemi $fit1$ a $fit2$, číslo 7 zastupuje průsečíky mezi funkcemi $fit1$ a $fceY$ a číslo 8 odpovídá průsečíkům mezi funkcemi $fit1$ a $fceY$. Výstupem funkce je hotová tabulka, pro zobrazení v prostředí funkce *Manipulate[],*

```
getTableOfIntersections[prusecikyFit12_, prusecikyFit1Y_, prusecikyFit2Y_, zobrazeni_] :=
 Module[{tabulka, nadpis = "Tabulka průsečíků"},
  tabulka = {If[MemberQ[zobrazeni, 6] && MemberQ[zobrazeni, 7] && MemberQ[zobrazeni, 8],
              {nadpis, SpanFromLeft, SpanFromLeft},
              If[ (MemberQ[zobrazeni, 6] && MemberQ[zobrazeni, 7]) ||
        (MemberQ[zobrazeni, 6] && MemberQ[zobrazeni, 8]) ||
        (MemberQ[zobrazeni, 7] && MemberQ[zobrazeni, 8]),
                    {nadpis, SpanFromLeft},
                    {nadpis}
               ]
              ],
         {}};
  If[MemberQ[zobrazeni, 6],
   AppendTo[tabulka[[2]],
    Grid[Prepend[Prepend[prusecikyFit12, {"x", "y"}],
      {"fit 1 a fit 2 " Style["●", Purple], SpanFromLeft}], Frame → All]]
  ];
  If[MemberQ[zobrazeni, 7],
   AppendTo[tabulka[[2]],
    Grid[Prepend[Prepend[prusecikyFit1Y, {"x", "y"}],
      {"fit 1 a fce y " Style["●", Orange], SpanFromLeft}], Frame → All]]
  ];
  If[MemberQ[zobrazeni, 8],
   AppendTo[tabulka[[2]],
    Grid[Prepend[Prepend[prusecikyFit2Y, {"x", "y"}],
      {"fit 2 a fce y " Style["●", Brown], SpanFromLeft}], Frame → All]]
  ];
  Return[Grid[tabulka, Frame → All, Alignment → {Center, Top}]];
 ]
```

Obr. 4. Funkce getTableOfIntersections[]

3.1.5 Funkce *getExtrems[]*

Funkce *getExtrems[]* je vytvořena pro hledání extrémů funkcí, hledá globální maximum a minimum na zadaném intervalu. Parametry $typ1$ a $typ2$ jsou nutné, aby funkce rozpoznala, že je zadána logaritmická funkce a v tom případě se dolní hranice intervalu pro hledání minima nastaví na hodnotu 10^-6 neboť použitá funkce *FindMinimum[]* pro menší hodnoty vrací chybovou zprávu. Parametry $fitFce1$, $fitFce2$ a $fceY$ slouží pro vložení funkcí k hledání jejich extrémů. Parametry $rozsahExtremuOd$ a $rozsahExtremuDo$ udávají interval na kterém je požadováno najít extrémy. Funkce *FindMaximum[]* a *FindMinimun[]* vrací souřadnice hledaného bodu, jen se mírně se poupraví na vektor a následně se z vektoru postaví výstupní matice.

```
getExtrems[typ1_, typ2_, fitFce1_, fitFce2_, fceY_, rozsahExtremuOd_, rozsahExtremuDo_] :=
 Module[{fitFce1Min, fitFce2Min, fceYMin, fitFce1Max, fitFce2Max, fceYMax},
  If[typ1 == 4,
   fitFce1Min = FindMinimum[{fitFce1[x], 10^-6 < x ≤ rozsahExtremuDo}, {x, Reals}],
   fitFce1Min = FindMinimum[{fitFce1[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}]
  ];
  fitFce1Min = {fitFce1Min[[2, 1, 2]], fitFce1Min[[1]]};
  fitFce1Max = FindMaximum[{fitFce1[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}];
  fitFce1Max = {fitFce1Max[[2, 1, 2]], fitFce1Max[[1]]};
  If[typ2 == 4,
   fitFce2Min = FindMinimum[{fitFce2[x], 10^-6 < x ≤ rozsahExtremuDo}, {x, Reals}],
   fitFce2Min = FindMinimum[{fitFce2[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}]
  ];
  fitFce2Min = {fitFce2Min[[2, 1, 2]], fitFce2Min[[1]]};
  fitFce2Max = FindMaximum[{fitFce2[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}];
  fitFce2Max = {fitFce2Max[[2, 1, 2]], fitFce2Max[[1]]};
  fceYMin = FindMinimum[{fceY[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}];
  fceYMin = {fceYMin[[2, 1, 2]], fceYMin[[1]]};
  fceYMax = FindMaximum[{fceY[x], rozsahExtremuOd ≤ x ≤ rozsahExtremuDo}, {x, Reals}];
  fceYMax = {fceYMax[[2, 1, 2]], fceYMax[[1]]};
  Return[{fitFce1Min, fitFce2Min, fceYMin, fitFce1Max, fitFce2Max, fceYMax}];
 ]
```

Obr. 5. Funkce getExtrems

3.1.6 Funkce *getTableOfExtrems[]*

Funkce *getTableOfExtrems[]* zajišťuje vizuální interpretaci tabulky extrémů. V parametru $extremy$ přebírá matici se souřadnicemi získanou z funkce *getExtrems[]* kterou zobrazí ve tvaru specifikovaném obsahem řídícího vektoru získaného parametrem $zobrazeni$. Základem je matice s počtem sloupců dle požadovaného počtu funkcí pro které je třeba zobrazit extrémy, která obsahuje pouze hlavní nadpis. Následně je matice doplněna tabulkami patřičných extrémů pro zvolené funkce. V případě že se v řídícím vektoru nachází číslo 9, do matice se doplní tabulka s minimem a maximem funkce $fit1$, pro číslo 10 je připojena tabulka s extrémy funkce $fit2$ a číslo 11 v řídícím vektoru zaručí přidání tabulky se souřadnicemi extrémů funkce $fceY$. Hotovou matici vrací funkce *getTableOfExtrems[]* jako tabulku.

```
getTableOfExtrems[extremy_, zobrazeni_] := Module[{tabulka, nadpis = "Tabulka extrémů"},
  tabulka = {If[MemberQ[zobrazeni, 9] && MemberQ[zobrazeni, 10] && MemberQ[zobrazeni, 11],
             {nadpis, SpanFromLeft, SpanFromLeft},
             If[(MemberQ[zobrazeni, 9] && MemberQ[zobrazeni, 10]) ||
       (MemberQ[zobrazeni, 9] && MemberQ[zobrazeni, 11]) ||
       (MemberQ[zobrazeni, 10] && MemberQ[zobrazeni, 11]),
                  {nadpis, SpanFromLeft}, {nadpis}]
             ],
        {}};
  If[MemberQ[zobrazeni, 9], AppendTo[tabulka[[2]],
    Grid[{{"Extrémy fit 1 " Style["●", Black], SpanFromLeft, SpanFromLeft, SpanFromLeft},
          {"Minimum", SpanFromLeft, "Maximum", SpanFromLeft},
          {"x", "y", "x", "y"},
          {extremy[[1, 1]], extremy[[1, 2]], extremy[[4, 1]], extremy[[4, 2]]}
     }, Frame → All]
   ]];
  If[MemberQ[zobrazeni, 10], AppendTo[tabulka[[2]],
    Grid[{{"Extrémy fit 2 " Style["●", Pink], SpanFromLeft, SpanFromLeft, SpanFromLeft},
          {"Minimum", SpanFromLeft, "Maximum", SpanFromLeft},
          {"x", "y", "x", "y"},
          {extremy[[2, 1]], extremy[[2, 2]], extremy[[5, 1]], extremy[[5, 2]]}
     }, Frame → All]
   ]];
  If[MemberQ[zobrazeni, 11], AppendTo[tabulka[[2]],
    Grid[{{"Extrémy fce y " Style["●", Cyan], SpanFromLeft, SpanFromLeft, SpanFromLeft},
          {"Minimum", SpanFromLeft, "Maximum", SpanFromLeft},
          {"x", "y", "x", "y"},
          {extremy[[3, 1]], extremy[[3, 2]], extremy[[6, 1]], extremy[[6, 2]]}
     }, Frame → All]
   ]];
  Return[Grid[tabulka, Frame → All, Alignment → {Center, Top}]];]
```

Obr. 6. Funkce getTableOfExtrems[]

3.1.7 Funkce *program[]*

Funkce *program[]* tvoří jádro celého programu, v jejím těle probíhají veškeré výpočty a generuje se grafický výstup při volání funkcí popsaných v předchozích odstavcích. Vstupními parametry jsou výstupy ovládacích prvků umístněných ve funkci *Manipulate[]*. Na začátku se sestaví funkce fitující vstupní data spuštěním funkce *getFitFunction[]*. Následně proběhne kontrola, zda uživatelem zadaná funkce $funkceY$ využívá některou z nalezených funkcí $fitFce1$ nebo $fitFce2$ a sestaví se z textového vstupu výsledná $fceY$. Dalším krokem je získání průsečíků funkcí pomocí funkce *getIntersections[]* a extrémů voláním funkce *getExtrems[]*. Grafický výstup se sestavuje do tabulky o čtyřech řádcích, kde první řádek zobrazuje požadovaný graf, druhý tvoří mezeru mezi prvním a třetím řádkem, do

kterého se v případě že řídící vektor zobrazování obsahuje čísla 6,7 nebo 8 vloží tabulka se souřadnicemi zvolených průsečíků a čtvrtý řádek za podmínky že se ve vektoru $zobrazeni$ nachází čísla 9,10 nebo 11 se vyplní tabulkou s extrémy pro zvolené funkce. Výsledný graf se vytváří pomocí funkce *Show[]*, jenž dokáže překrývat jednotlivé grafy, čímž vytvoří jeden, který může obsahovat libovolně zvolené funkce. Volbu grafů, které se mají překrývat, zajišťuje blok podmínek *If[]* řízených vektorem $zobrazeni$.

Číselná reprezentace volby:

1. - vykreslení funkce $fit1$
2. - vykreslení funkce $fit2$
3. - vykreslení vstupních dat $data1$
4. - vykreslení vstupních dat $data1$
5. - vykreslení funkce $fceY$
6. - vykreslení průsečíků funkcí $fit1$ a $fit2$
7. - vykreslení průsečíků funkcí $fit1$ a $fceY$
8. - vykreslení průsečíků funkcí $fit2$ a $fceY$
9. - vykreslení extrémů funkce $fit1$
10. - vykreslení extrémů funkce $fit2$
11. - vykreslení extrémů funkce $fceY$

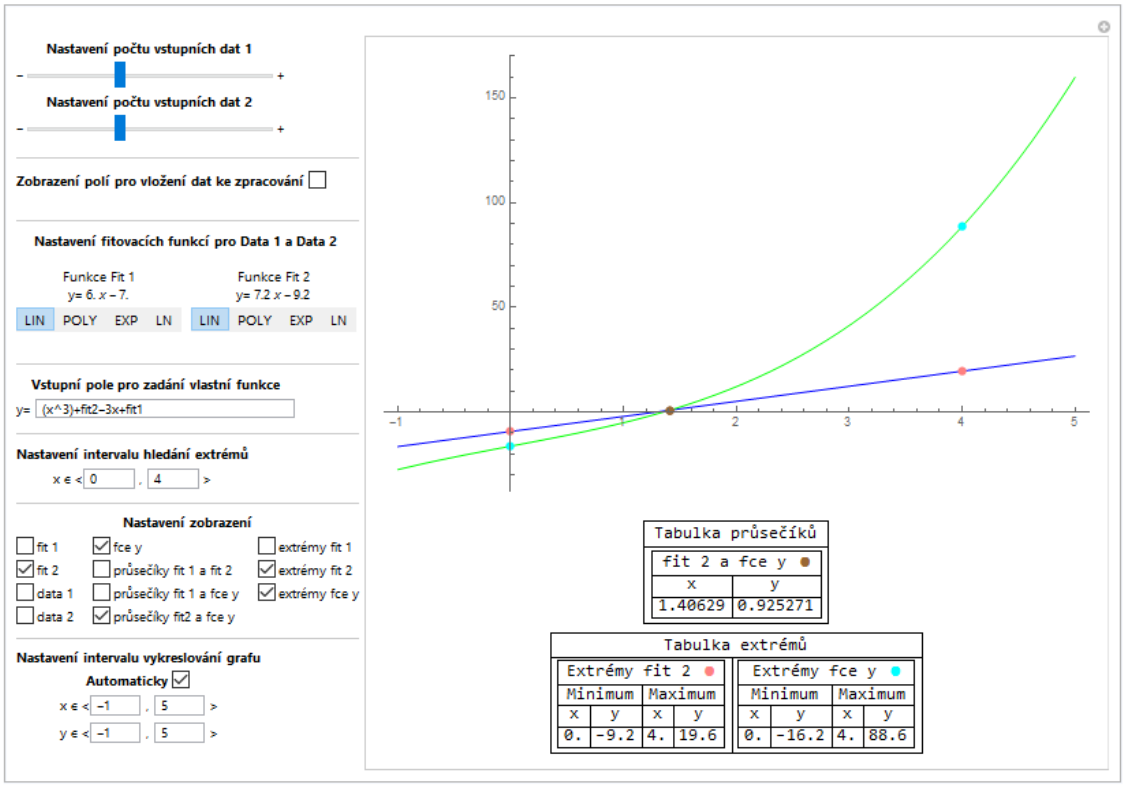
Volby jsou v uživatelském rozhraní funkce *Manipulate[]* reprezentovány zaškrtávacími políčky.

```
program[funkceY_, data1_, data2_, typ1_, typ2_, stupenPolynomu1_, stupenPolynomu2_,
  rozsahExtremuOd_, rozsahExtremuDo_, vykresleniOd_, vykresleniDo_, zobrazeni_, osaYOd_,
  osaYDo_, vykresleni_] :=
 Module[{fceY, fitFce1, fitFce2, prusecikyFit12, prusecikyFit1Y, prusecikyFit2Y, extremy,
   prazdnyGraf, tabulka},
  fitFce1 = getFitFunction[typ1, data1, stupenPolynomu1];
  fitFce2 = getFitFunction[typ2, data2, stupenPolynomu2];
  fceY = StringReplace[funkceY, "fit1" → "(" <> ToString[fitFce1[x], StandardForm] <> ")"];
  fceY = StringReplace[fceY, "fit2" → "(" <> ToString[fitFce2[x], StandardForm] <> ")"];
  fceY = ToExpression[fceY];
  fceY = Function[{x}, Evaluate[fceY]];
  prusecikyFit12 = getIntersections[fitFce1, fitFce2];
  prusecikyFit1Y = getIntersections[fitFce1, fceY];
  prusecikyFit2Y = getIntersections[fitFce2, fceY];
  extremy = getExtrems[typ1, typ2, fitFce1, fitFce2, fceY, rozsahExtremuOd, rozsahExtremuDo];
  prazdnyGraf = Plot[{}, {x, vykresleniOd, vykresleniDo}];
  tabulka = Grid[{
     {Show[
       If[MemberQ[zobrazeni, 1], Plot[fitFce1[x], {x, vykresleniOd, vykresleniDo}, PlotStyle → Red],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 2], Plot[fitFce2[x], {x, vykresleniOd, vykresleniDo}, PlotStyle → Blue],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 3], ListPlot[data1, PlotStyle → {Red, PointSize[Large]}], prazdnyGraf],
       If[MemberQ[zobrazeni, 4], ListPlot[data2, PlotStyle → {Blue, PointSize[Large]}], prazdnyGraf],
       If[MemberQ[zobrazeni, 5], Plot[fceY[x], {x, vykresleniOd, vykresleniDo}, PlotStyle → Green],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 6], ListPlot[prusecikyFit12, PlotStyle → {Purple, PointSize[Large]}],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 7], ListPlot[prusecikyFit1Y, PlotStyle → {Orange, PointSize[Large]}],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 8], ListPlot[prusecikyFit2Y, PlotStyle → {Brown, PointSize[Large]}],
        prazdnyGraf],
       If[MemberQ[zobrazeni, 9], ListPlot[{extremy[[1]], extremy[[4]]},
         PlotStyle → {Black, PointSize[Large]}], prazdnyGraf],
       If[MemberQ[zobrazeni, 10], ListPlot[{extremy[[2]], extremy[[5]]},
         PlotStyle → {Pink, PointSize[Large]}], prazdnyGraf],
       If[MemberQ[zobrazeni, 11], ListPlot[{extremy[[3]], extremy[[6]]},
         PlotStyle → {Cyan, PointSize[Large]}], prazdnyGraf],
       If[vykresleni == 1, PlotRange → Automatic,
        PlotRange → {{vykresleniOd, vykresleniDo}, {osaYOd, osaYDo}}],
       ImageSize → Large]
     },
     {""},
     {If[MemberQ[zobrazeni, 6] || MemberQ[zobrazeni, 7] || MemberQ[zobrazeni, 8],
       getTableOfIntersections[prusecikyFit12, prusecikyFit1Y, prusecikyFit2Y, zobrazeni]
      ]},
     {If[MemberQ[zobrazeni, 9] || MemberQ[zobrazeni, 10] || MemberQ[zobrazeni, 11],
       getTableOfExtrems[extremy, zobrazeni]
      ]}
    }];
  Return[tabulka];
 ]
```

Obr. 7. Funkce program[]

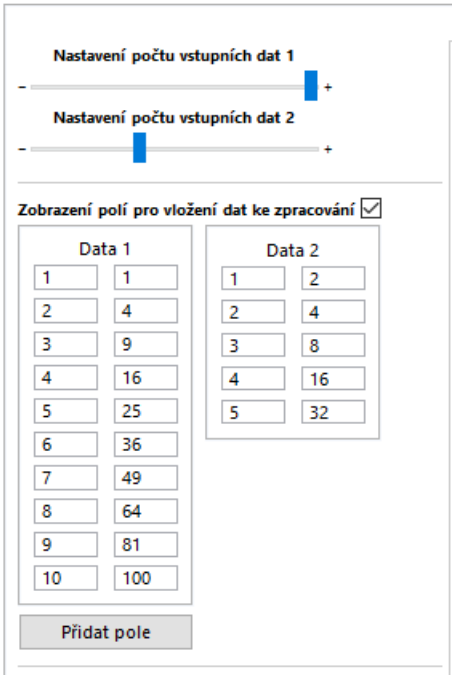
3.2 Popis uživatelského rozhraní

Panel aplikace je rozdělen do dvou částí, napravo se nachází grafický výstup (zobrazení grafu funkcí a případně tabulky souřadnic průsečíků a extrémů), vlevo jsou umístněny ovládací prvky.



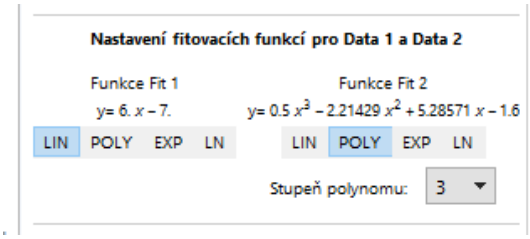
Obr. 8. Uživatelské rozhraní vytvořené aplikace

Posuvníky *Nastavení počtu vstupních dat 1* a *2* slouží k nastavení velikosti tabulky pro vkládání dat a současně určují počet zpracovávaných dat (po zmenšení tabulky se se skrytými daty nepočítá). Pro snadnější orientaci mezi ovládacími prvky lze pomocí zaškrtávacího pole u nápisu *Zobrazení polí pro vložení dat ke zpracování* jednoduše tabulky pro vkládání dat skrýt (tato volba neovlivňuje výpočty). V případě že přednastavený rozsah velikosti tabulek nebude uživateli dostačující, pod tabulkou u které bude rozsah nastaven na maximum, se zobrazí tlačítko pro přidání nového páru vstupních polí. Po aktivaci tlačítka se zvětší nastavitelný rozsah u dané tabulky o jednu hodnotu.



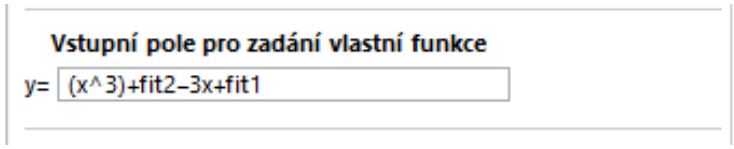
Obr. 9. Nastavení vstupních dat

V bloku ovládacích prvků s popisem *Nastavení fitovacích funkcí pro Data 1 a Data 2* lze jednoduše zvolit, jakou funkcí se vložená data proloží. Na výběr je funkce lineární, polynomická, exponenciální, nebo logaritmická, u polynomické je možno zvolit stupeň polynomu funkce mezi druhým až pátým stupněm.



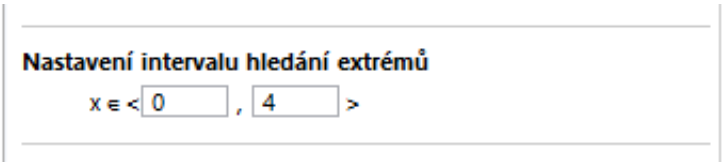
Obr. 10. Nastavení fitovacích funkcí pro Data 1 a Data 2

Blok *Vstupní pole pro zadání vlastní funkce* umožňuje zadat vlastní funkci, která může pracovat s funkcemi získanými fitováním vstupních dat, do rovnice funkce je možno zapsat *fit1* pro dosazení funkce Fit 1 a nebo *fit2* pro dosazení funkce Fit 2.



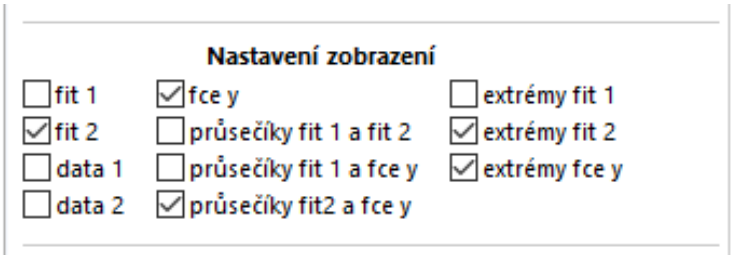
Obr. 11. Vstupní pole pro zadání vlastní funkce

Blok *Nastavení intervalu hledání extrémů* slouží k nastavení intervalu pro hledání extrémů. Funkce pro hledání extrémů vrací pouze globální extrémy, pro nalezení konkrétního lokálního extrému je nutno upravit tento interval, aby funkce vrátila souřadnice požadovaného extrému.



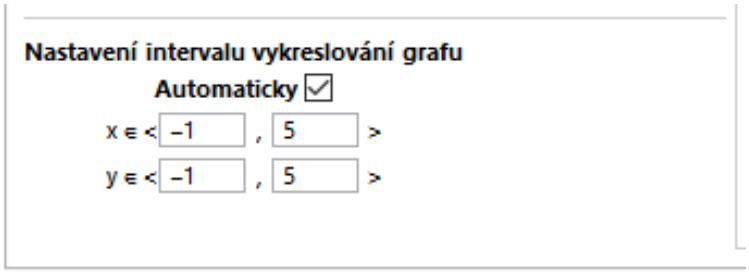
Obr. 12. Nastavení intervalu hledání extrémů

Blok *Nastavení zobrazení* zaškrtávací políčka reprezentují zobrazení dané funkce nebo dat v grafu, u průsečíků a extrémů se pod grafem zobrazuje i tabulka s jejich souřadnicemi.



Obr. 13. Nastavení zobrazení

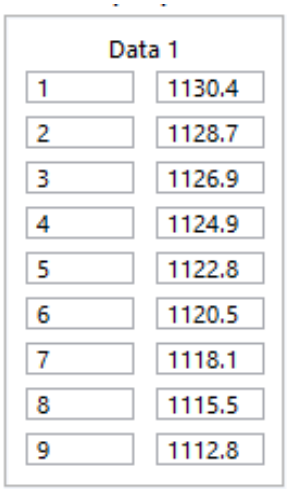
Blok *Nastavení intervalu vykreslování grafu* umožňuje volit mezi automatickým vykreslením grafu a manuálním nastavením rozsahů os. Po zrušení volby *Automaticky* je možné přepisovat intervaly určující rozsah grafu.



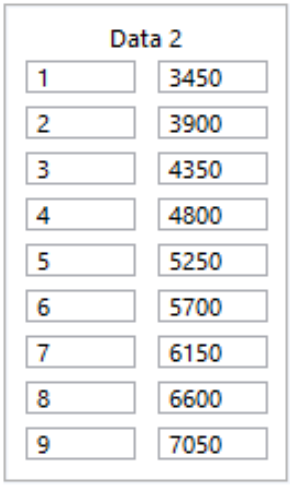
Obr. 14. Nastavení intervalu vykreslování grafu

3.3 Řešení příkladu

Funkce poptávky a celkových nákladů jsou zadány tabulkami dat viz *Obr. 15* a *Obr. 16.*

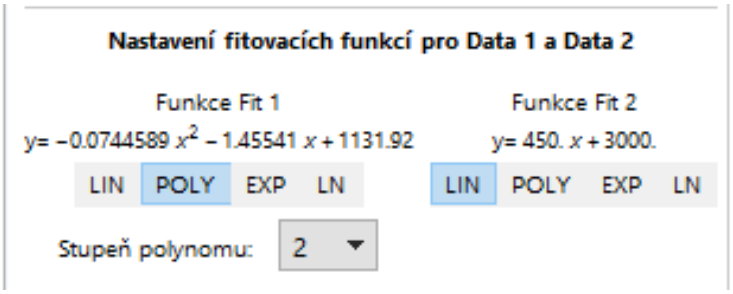


Obr. 15. Data reprezentující funkci poptávky



Obr. 16. Data reprezentující funkci celkových nákladů

V prostředí aplikace se nastaví požadované fitovací funkce viz *Obr. 17.*

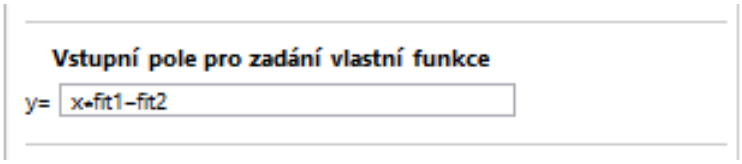


Obr. 17. Nastavení fitovacích funkcí

Funkce zisku $TP(x)$ je obecně dána vztahem:

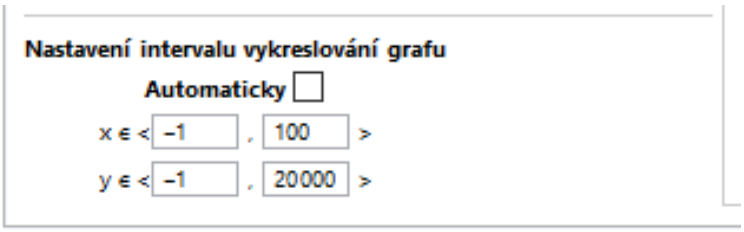
$$TP(x) = xp - TC(x)$$

Cenu p udává funkce poptávky reprezentovaná funkcí *Fit 1* a funkce celkových nákladů je reprezentována funkcí *Fit 2.* Výsledný vztah $TP(x) = x \cdot fit1 - fit2$ se v nastavení aplikace vloží do pole pro zadání vlastní funkce *Obr. 18*.



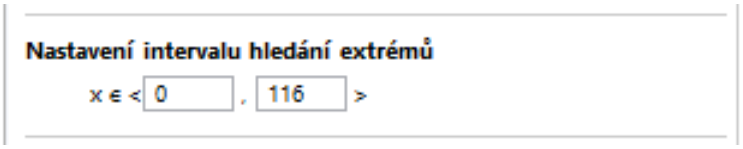
Obr. 18. Pole pro zadání vlastní funkce

Automatická volba vykreslení grafu nedokáže vždy odhadnout ideální rozsah zobrazení, proto je pro tento příklad vhodné nastavit rozsahy ručně dle *Obr. 19*.



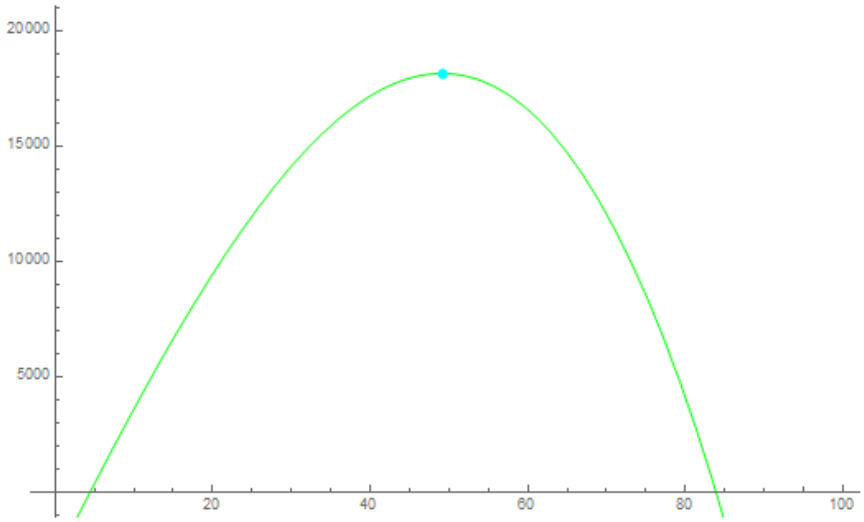
Obr. 19. Nastavení rozsahů vykreslení grafu

Pro správné nalezení maxima funkce je třeba nastavit interval pro hledání extrémů. Pro množství výrobků x platí, že $x \geq 0$. Pro cenu p danou funkcí poptávky platí také, že musí být nezáporná, rovnice fitované funkce má průsečík s osou x v bodě $x \doteq 116$. Hledání extrémů pro proměnnou x má tedy smysl v intervalu $0 \leq x \geq 116$.



Obr. 20. Nastavení intervalu hledání extrémů

Zobrazením maxima funkce *Fce y* jenž reprezentuje funkci zisku, lze ze souřadnic bodu vyčíst, že maximálního zisku se dosáhne při množství přibližně 50 kusů výrobků (x-ová souřadnice) a hodnota zisku bude přibližně 18160 Kč.



Tabulka extrémů			
Extrémy fce y			
Minimum		Maximum	
x	y	x	y
0.	-3000.	49.1195	18160.

Obr. 21. Graf funkce zisku s nalezeným maximem

ZÁVĚR

Hlavním výsledkem této práce je interaktivní aplikace, která dokáže vstupní data proložit funkcí a umožňuje snadnou vizualizaci zvolených funkcí. Ovládání je velmi jednoduché a po přečtení kapitoly 3.2 by měl být aplikaci schopen ovládat každý.

Aplikace by měla najít uplatnění především v řešení úloh na minimalizaci nákladů, maximalizaci zisku a vztahy mezi nabídkou a poptávkou, kdy uživatel má k dispozici data a je třeba nalézt funkci, jenž je těmito daty reprezentována. Pro všechny zadané funkce, je k dispozici hledání extrémů a lze zjistit vzájemné průsečíky mezi funkcemi.

Aplikace by měla primárně sloužit pro práci s ekonomickými funkcemi, ale užitečná by mohla být v jakémkoliv oboru při hledání extrémů funkcí nebo fitování dat.

SEZNAM POUŽITÉ LITERATURY

[1] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Zlín: UTB FAI, 2006. 122 s. ISBN 80-7318-510-5

[2] WOLFRAM, Stephen. *The mathematica book. 5th ed. Champaign*, IL: Wolfram Media, 2003. 1464 s. ISBN 1579550223.

[3] Wolfram Language andSystem documentation center [online]. 2017 [cit. 2017-02-03]. Dostupné z WWW: [http://reference.wolfram.com/language/].

[4] ALLEN, R.G.D. *Matematická ekonomie.* Praha: ACADEMIA, 1971. 784 s. ISBN nemá.

[5] SEKERKA, Bohuslav. *Mikroekonomie – matematické a kvantitativní základy*. Profess Consulting, 2002. 361 s. ISBN 80-7259-030-8

[6] KŘENEK, Josef a OSTRAVSKÝ, Jan. *Diferenciální a integrální počet funkce jedné proměnné s aplikacemi v ekonomii*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006. ISBN 80-7318-530-x.

[7] ZIMKA, Rudolf. *Matematika I s aplikáciami v ekonómii*. Zvolen: Matcentrum, 1999. ISBN 80-9680-572-x.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH