Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Finanční matematika na internetu s podporou formátu cdf

Vypracoval: Leoš Glaser
Vedoucí práce: doc. RNDr. Tomáš Mrkvička, Ph.D.

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Leoš GLASER**
Osobní číslo:	**E18304**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Finanční matematika na internetu s podporou formátu cdf**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvoření aplikací pro finanční kalkulace s podporou formátu cdf. Tento formát umožňuje přímo do dokumentu zadávat vstupní parametry výpočtu. Díky tomu se vytvořené dokumenty stanou interaktivní a dostupné široké veřejnosti. Dalším cílem práce je uveřejnění těchto dokumentů na internetu.

Metodický postup:

1. Studium literatury – 1. semestr.
2. Vytvoření cdf dokumentů – 2. semestr.
3. Zpracování výsledků – 3. semestr.
4. Závěr a doporučení.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Cipra, T. (2015). *Praktický průvodce finanční a pojistnou matematikou.* Praha: Ekopress, s.r.o.
2. Computable Document Format. Wolfram. [online]. 2020. Dostupné z WWW: <https://www.wolfram.com/cdf/>.
3. Zima, P. & Brown, L. R. (1996). *Schaum's Outline of Theory and Problems of Mathematics of Finance.* New York: McGraw-Hill Companies, Inc.
4. Další odborná časopisecká a knižní literatura dle konkrétního zaměření práce.

Vedoucí bakalářské práce: **doc. RNDr. Tomáš Mrkvička, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **17. ledna 2020**
Termín odevzdání bakalářské práce: **16. dubna 2021**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 4. března 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne

_____

Leoš Glaser

Poděkování

Děkuji svému vedoucímu doc. RNDr. Tomášovi Mrkvičkovi, PhD. za jeho rady a připomínky při zpracovávání této práce. Dále bych rád poděkoval Bc. Zdeňkovi Heřmanovi za vytvoření domény na serveru fakulty. Rovněž děkuji své rodině za jejich podporu během studia.

Obsah

Úvod

Jedním z trendů v oblasti vzdělávání je snaha o co největší digitalizaci a využití moderních výpočetních technologií ve výuce. Přestože již dlouhou dobu je k dispozici nemálo nástrojů pro tvorbu materiálů podporujících interaktivitu, většina materiálů zůstává ve statické podobě. Toto byla motivace, která vedla společnost Wolfram Research k vytvoření vlastního dynamického interaktivního formátu *Computable Document Format* (CDF).

Formát CDF umožňuje vytvářet a sdílet dokumenty s uživatelským rozhraním pro zadávání parametrů výpočtu, přičemž další zobrazovaný obsah v dokumentu dynamicky reaguje na změny jednotlivých parametrů. Díky CDF tedy lze sdílet s ostatními různé kalkulátory, modely, applety, které díky svojí interaktivní povaze mohou usnadnit pochopení dané problematiky a ulehčit výpočty. Tyto CDF dokumenty si kdokoliv může otevřít ve zdarma dostupném softwaru *CDF Player*.

Cílem této práce je vytvoření CDF dokumentů pro kalkulace z oblasti finanční matematiky. Práce je rozdělena na tři části. První část je teoretická a představuje vybrané produkty společnosti Wolfram Research, důraz je kladen zejména na software *Mathematica*. V této části je rovněž popsán formát CDF, jeho výhody, nevýhody a příklady užití. Druhá část je zaměřena prakticky a zabývá se vývojem jednotlivých dokumentů pro vybrané oblasti finanční matematiky. Ve třetí části jsou popsány možnosti sdílení CDF dokumentů na internetu. V závěru práce jsou shrnuty poznatky a doporučena témata, která by na tuto práci mohla navazovat.

1 Wolfram Research

1.1 Wolfram Mathematica

1.1.1 Základní informace

Software Wolfram Mathematica je jeden ze světově nejznámějších nástrojů pro vědecko-technické výpočty. Jedná se o hlavní a nejznámější produkt společnosti Wolfram Research a je primárně určený pro oblast výzkumu, vývoje a vzdělávání. První verze, která vyšla v roce 1988, byla zaměřena na oblast matematiky a obsahovala přes 500 vestavěných funkcí. Postupným vývojem a vylepšováním se však výrazně rozšířil počet oborů, kde lze Mathematicu využít pro řešení mnoha rozmanitých problémů. V současnosti může Mathematica nalézt uplatnění například v oblastech matematiky, informatiky, fyziky, chemie, ekonomie, robotiky a v dalších oblastech. Nejnovější verze č. 12 byla vydána v roce 2019 a obsahuje více než 6 tisíc vestavěných funkcí. Přestože se software rozvíjí více než 30 let, je navržen a rozšiřován tak, aby byla zajištěna co největší zpětná kompatibilita s předchozími verzemi i s ostatními produkty společnosti Wolfram Research. (*Wolfram Mathematica: The world's definitive system for modern technical computing*, n.d.)

Mathematica existuje ve dvou variantách: *desktop* a *online*. Mathematica Desktop se instaluje na počítač jako samostatný program, zatímco Mathematica Online běží ve Wolfram Cloudu a k jejímu užívání stačí pouze internetový prohlížeč a aktivní Wolfram Cloud účet. Verze se mezi sebou funkčně zásadně neliší, avšak rozdíly existují. Desktop verze nabízí více možností pro zadávání vstupu od uživatele, rychlejší výpočty díky faktu, že výpočetní jádro běží v lokálním prostředí a v důsledku má desktopová verze i plynulejší reakce zobrazovaných prvků na změny parametrů výpočtu. Online verze naopak umožňuje některé věci, které jsou specifické pro webové prostředí, například vložení YouTube videa do vytvářené aplikace. (Hastings, 2015)

1.1.2 Struktura

Mathematica sestává ze dvou částí (Torrence & Torrence, 2019):

1. *Kernel*, neboli výpočtové jádro. Kernel přijímá příkazy zadané ve front end části, provede příslušné výpočty a výsledky zašle zpět na front end. Kernel může být spuštěn na samostatném stroji, lze vytvářet i subkernely a docílit tak paralelizaci výpo-

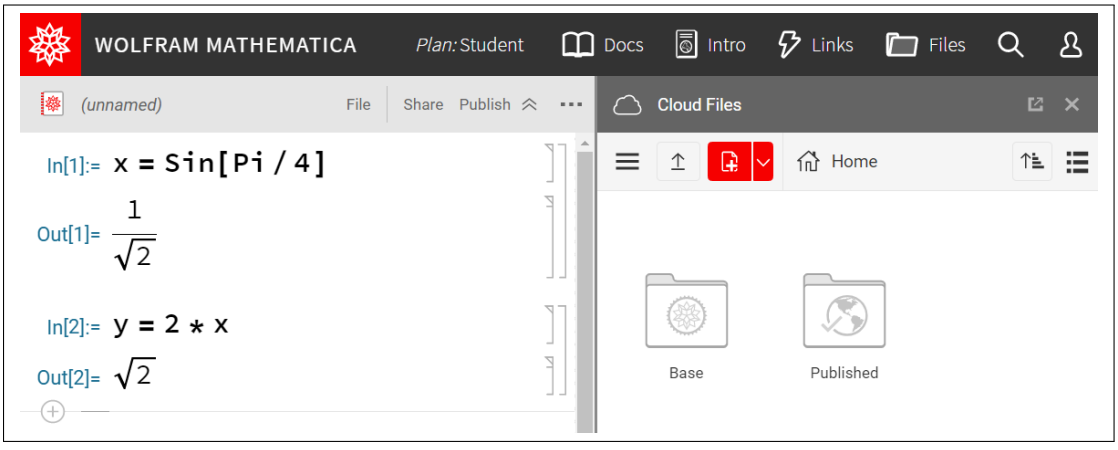
čtů. Kernel je napsán ve výkonném programovacím jazyce C a obsahuje mnoho tzv. vestavěných (anglicky „built-in“) funkcí. (Shifrin, 2008)

2. *Front end*, neboli grafické uživatelské rozhraní. Tato část zajišťuje komunikaci s kernelem, zobrazování výstupů přijatých z kernelu, ovládacích prvků, nastavování vizuálních stylů textu a další věci.

   Základem front end prostředí jsou notebooky. Notebooky jsou strukturované dokumenty poskytující uživateli plně interaktivní prostředí. (*Notebook Basics*, n.d.) S kernelem lze komunikovat i skrze textové prostředí, nicméně v takovém případě není možné používat grafické ani dynamické prvky (např. vykreslení grafu funkce, ovládací tlačítka). (*Using a Text-Based Interface*, n.d.)

Notebooky se skládají z buněk různých typů, například buňky pro vstup, výstup, obyčejný text, nadpisy. Buňky je možné do sebe vnořovat, skrývat je, vyhodnocovat či naopak jejich vyhodnocování zakázat, kopírovat je anebo obsah buněk zveřejňovat na internetu. Vstupní (resp. výstupní) buňky jsou označené „`In[n]:=`“ (resp. „`Out[n]=`“), kde `n` je pořadové číslo buňky. Na obrázku 1 je ukázka jednoduchého notebooku se dvěma vstupními buňkami a jim odpovídajícími výstupními buňkami.

Obrázek 1: Jednoduchý notebook



Zdroj: autor

Jednou z výhod vyčlenění výpočtů do samostatného kernelu je to, že nastane-li při výpočtech chyba nebo nějaký nežádoucí stav, není nutné restartovat celý program Mathematica. Stačí pouze restartovat kernel a na daný notebook toto nemá vliv.

1.1.3 Wolfram Language

Notebooky se programují pomocí jazyka *Wolfram Language*. Jedná se o symbolický jazyk zaměřený zejména na funkcionální programování, tj. programy jsou chápány jako sekvence funkcí, které mohou být různě skládány. Například výraz `2 + 3*x` je ve skutečnosti složení funkcí pro sčítání a násobení: „`Plus[2, Times[3,x]]`." Jazyk však podporuje i jiná programovací paradigmata, například procedurální a strukturované programování.

Nejzásadnějším rysem jazyka je právě fakt, že vše je symbolický výraz (anglicky „symbolic expression") a vešekré výpočty tedy mohou probíhat čistě symbolicky. Symbolické výrazy se skládají z tzv. *atomů*, což jsou nejčastěji čísla, textové řetězce a symboly. (Wolfram, 2017, p. 205-206) V příkladu výrazu na předchozí stránce jsou číselnými atomy `2`, `3` a symbolickým atomem `x`.

Symbolické výrazy jsou kernelem vyhodnocovány, přičemž vyhodnocování funguje postupným aplikováním série definicí až do momentu, kdy již žádná definice není dostupná. Je-li třeba zadán výraz `x = 8; x + f[3] + 4 f[3]`, tak při jeho vyhodnocování se použijí dvě definice: jedna definuje symbol `x` a druhá je vestavěná definice pro zjednodušování součtu výrazů. Není ale zadána definice pro výraz `f[3]` a proto výsledkem vyhodnocení celého původního výrazu je `8 + 5 f[3]`.

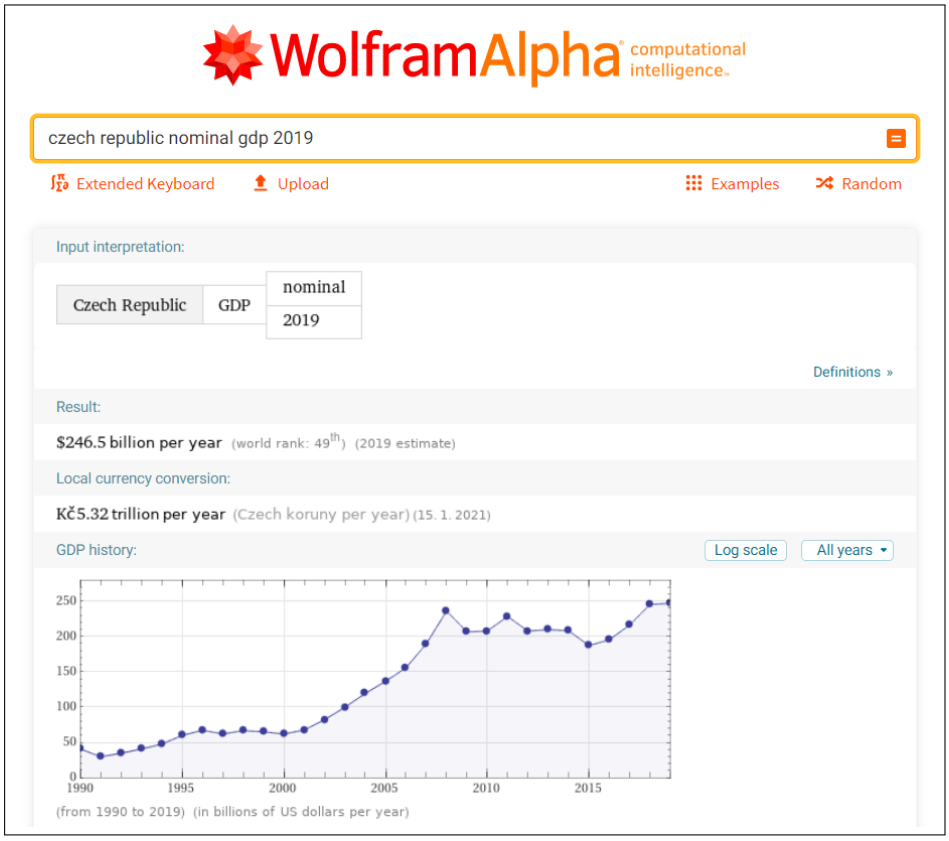
Příklady užití jazyka pro vývoj aplikací jsou popsány v praktické části této práce.

1.2 Wolfram Alpha

Wolfram Alpha je webová aplikace, konkrétněji odpovídací stroj (anglicky „answer engine") inspirovaný Mathematicou a napsaný v jazyce Wolfram. Hlavním cílem je snaha zajistit, aby *všechny systematické znalosti byly okamžitě vypočítatelné a přístupné všem* (*About Wolfram|Alpha*, n.d.).

Aplikace byla spuštěna v roce 2009 a je zdarma dostupná na adrese `wolframalpha.com`. Do zadávacího pole se píší dotazy, přičemž aplikace umí rozpoznat a vyhodnotit i dotazy zadané v přirozeném jazyce. Na obrázku 2 je ukázka dotazu na HDP České republiky za rok 2019. V sekci „Input interpretation" lze vidět, jak aplikace daný dotaz interpretovala. Je možné se proklikat přímo až k výrazu, pomocí kterého aplikace vypočítala odpověď. Výraz je možné zadat do Mathematicy a takto by se došlo ke stejnému výsledku, jaký zobrazil Wolfram Alpha (v tomto konkrétním případě `$246.5*10^11 per year`).

Obrázek 2: Odpověď Wolfram Alpha



Zdroj: autor

1.3 Formát CDF

1.3.1 Základní popis

Computable Document Format (do češtiny lze přeložit jako „formát spočitatelného dokumentu“) je podobný notebookům. Představen byl v roce 2011. Hlavní motivací pro CDF je usnadnění komunikace myšlenek díky interaktivní povaze dokumentu. Narozdíl od statických dokumentů (např. ve formátu PDF) je možné v CDF dokumentech zadávat a měnit parametry výpočtu a dokument na toto reaguje změnou zobrazovaného výstupu. Uživatelé si tedy sami mohou zkoušet výpočty s různým nastavením, což jim může usnadnit pochopení dané problematiky. (*Why Use the Computable Document Format (CDF)?*, n.d.)

Zatímco s notebooky se dá plnohodnotně pracovat jedině v Mathematice, na otevření a interakci s CDF dokumenty je možné použít rovněž Mathematicu, anebo volně dostupný software *Wolfram Player*. V něm jdou otevřít i notebooky, ale není možné vyhodnocovat buňky.

1.3.2 Možná užití CDF

CDF lze využít pro tvorbu aplikací zaměřených na demonstraci specifických problémů, ale dále i k tvorbě interaktivních prezentací, automatizovaných reportů, dynamických článků anebo celých učebních textů. Příkladem poslední zmíněné možnosti je elektronická učebnice matematické analýzy vytvořená britskou společností Pearson Education. Učebnice obsahuje více než 600 interaktivních demonstrací ve formě CDF aplikací. (Briggs, Cochran, & Gillett, n.d.) Zatímco v běžném statickém dokumentu bývá k dostatečně detailnímu vysvětlení určité věci (např. aproximace plochy pod křivkou pomocí Riemannových součtů) mnohdy potřebné využít vícero obrázků, v interaktivním textu lze vystačit třeba i s pouze jediným vhodně navrhnutým CDF appletem.

1.4 Wolfram Demonstrations Project

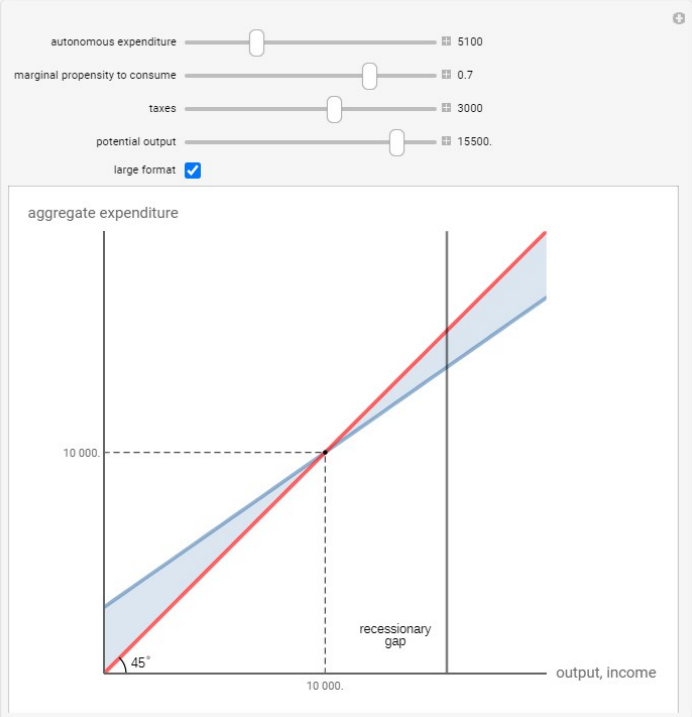
1.4.1 Demonstrace

Wolfram Demonstrations je webový portál obsahující sbírku více než 12 tisíc interaktivních demonstrací (appletů, aplikací), které si lze stáhnout ve formě notebooku nebo jako CDF dokument. Některé demonstrace jsou optimalizovány pro lokální používání na počítači, většina je však vhodná i pro užití na internetu. Rozdílem je rychlost odezvy, která je vyšší při lokálním používání demonstrací. Demonstrace jsou organizovány podle oborů a pokrývají mnoho konkrétních problémů a příkladů z matematiky, fyziky, biologie, ekonomie, strojního inženýrství, teorie her a mnoha dalších oborů.

Demonstrace obsahují jednoduché uživatelské rozhraní ovládající výstup, který je nejčastěji ve formě textového pole, grafu nebo tabulky, ale lze najít i ukázky s pokročilejší vizualizací, například na obrázku 4 je simulace porovnávající klasický spalovací motor a motor s rozdělenými cykly.

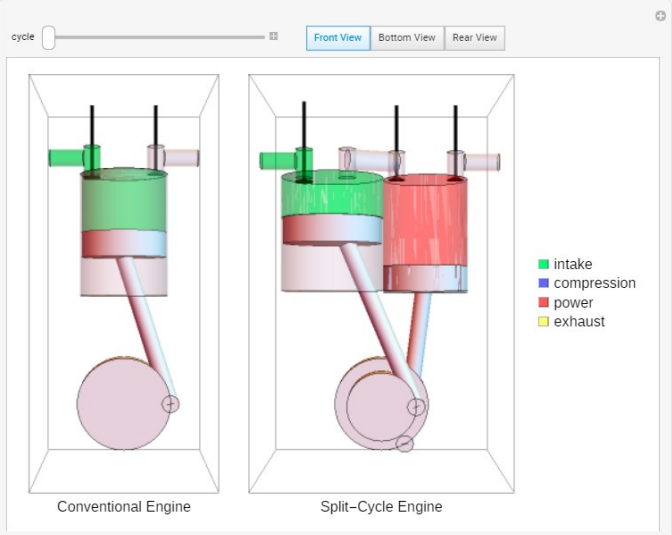
Demonstrace může do Wolfram Demonstrations nahrávat každý uživatel aplikace Wolfram Mathematica, který má na stránce projektu založený účet. U většiny demonstrací si lze stáhnout i jejich zdrojové kódy. Zveřejnění kódu však není nutné.

Obrázek 3: Demonstrace keynessiánského kříže



Zdroj: (Maclachlan, 2011)

Obrázek 4: Demonstrace spalovacích motorů



Zdroj: (Yuan, 2014)

1.4.2 Pravidla pro tvorbu demonstrací

Počet oborů demonstrací je veliký, podstata jednotlivých vizualizovaných problémů se obor od oboru a příklad od příkladu značně liší a každý autor má svůj specifický styl. Proto jsou na stránkách projektu uvedena doporučení a příkazy, jejichž cílem je do určité míry ujednotit styl, jak mají být demonstrace vyvíjeny a jak má výsledná demonstrace vypadat a fungovat.(*Author Guidelines*, n.d.)

Pro praktickou část této práce jsou relevantní zejména následující pravidla:

1. Popisky ovládacích prvků a další texty mají být v angličtině a mají být co nejvýstižnější.
2. Pouze vlastní jména začínají velkým písmenem, ostatní popisky začínají malým písmenem.
3. Ovládacích prvků by nemělo být příliš. Demonstrace by kvůli nim neměla být nepřehledná.
4. Nelze použít ovládací prvek typu `InputField` (jedná se o prvek fungující jako políčko formuláře, kam se dá vepsat vstup ve formě alfanumerických znaků).
5. Demonstrace musí obsahovat pouze jeden blok `Manipulate[]` (ten bude blíže popsán v praktické části práce)
6. Všechny proměnné používané v `Manipulate[]` musí být lokální.
7. Při manipulaci s ovládacími prvky by se velikost demonstrace i jejích ovládacích prvků neměla měnit.
8. Při nastavení ovládacích prvků do jejich krajních hodnot by neměla nastat žádná chyba nebo nežádoucí chování demonstrace.

Jelikož CDF dokumenty vytvářené v rámci této práce nebudou zveřejněny na Wolfram Demonstrations, nebudou výše uvedené pravidla dodržena na 100 %. Bod č. 1 nebude dodržen, jelikož vytvářené aplikace jsou určeny především pro podporu výuky finanční matematiky v českém jazyce. Ze stejného důvodu nebude zcela dodržen ani bod 2. Vzhledem k charakteru některých témat a zvoleného způsobu vizualizace bude místy porušen i bod 7. Každopádně však budou CDF dokumenty v praktické části vyvíjeny tak, aby odchylky od výše uvedených pravidel byly co nejmenší.

2 CDF dokumenty pro finanční matematiku

V následujících podkapitolách bude popsána tvorba CDF dokumentů týkajících se vybraných partií finanční matematiky. U každého tématu je nejdříve uvedena základní teorie, která je následně zapracována do CDF dokumentů.

2.1 Úročení

2.1.1 Jednoduché úročení

U jednoduchého úročení se úročí pouze základ, tj. nepočítají se úroky z úroků. Jednoduché úročení se nejčastěji používá v situacích, kdy doba půjčky není delší než 1 rok. (Cipra, 2015, str. 14) Vztahy pro jednoduché úročení:

$$A = P + I \tag{1}$$

$$I = P \cdot i \cdot t \tag{2}$$

kde

- A budoucí hodnota (zúročená částka)
- P základ (kapitál, jistina)
- I jednoduchý úrok
- i roční úroková míra
- t doba úročení (v letech)

Pro určení doby t se nejčastěji používají následující standardy:

- $30E/360$ (evropský standard, německá metoda): rok má 360 dní, měsíce 30 dní
- $ACT/360$ (francouzská/mezinárodní metoda): rok má 360 dní, každý měsíc skutečný počet dní
- $ACT/365$ (anglická metoda): rok i měsíc mají skutečný počet dní

CDF aplikace pro jednoduché úročení bude umět vypočítat budoucí i současnou hodnotu, úrokovu míru i dobu úročení, se kterou se bude počítat dle zvoleného standardu. Základem aplikace je příkaz `Manipulate[expr, cont]`, který má dva parametry: výraz `expr` a ovládací prvek parametru daného výrazu.

Při vytváření CDF dokumentu se nejprve napíšou ovládací prvky pro výběr neznámé, která bude počítána, a prvek ovládající počet desetinných míst zobrazovaných výsledků.

Kód 1: Základ CDF aplikace pro jednoduché úročení

```
Manipulate[
  x,
  Style["Jednoduché úročení", Bold],
  {{unknown, "P", "neznámá"}, {
    "P" -> "současná hodnota",
    "A" -> "budoucí hodnota",
    "i" -> "úroková míra",
    "t" -> "období"}
    },
{{precision, 2, "desetinná místa"}, 0, 10, 1,
 AppearanceElements -> None, Appearance -> "Labeled"},

 Initialization->(
 )
]
```

Zdroj: autor

Proměnná unknown má nastavenou počáteční hodnotu na "P". U obvou ovládacích prvků není nutné uvést jejich typ - Mathematica jej určí automaticky sama. Nyní se vytvoří ovládací prvky pro P,A, i a volbu standardu (kód 2).

Kód 2: Ovládací prvky P,A, i a standardu

```
{{P, 1, "současná hodnota"}, 1, 10^6,
 TrackingFunction -> (P = If[And[# > A, unknown =!= "A"], A, #]; &),
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "P"),
 AppearanceElements -> None},

{{A, 1, "budoucí hodnota"}, 1, 10^6,
 TrackingFunction -> (A = If[And[# < P, unknown =!= "P"], P, #]; &),
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "A"),
 AppearanceElements -> None},

{{i, 1, "úroková míra"}, 1, 100,
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "i"),
 AppearanceElements -> None},

{{tStandard, tStandard30E360, labeltStandard}, {
    tStandard30E360 -> "30E/360",
    tStandardACT360 -> "ACT/360",
    tStandardACT365 -> "ACT/365"
```

```
    },
  ControlType -> RadioButton,
  Enabled -> (unknown =!= "t"),
  AppearanceElements -> None}
```

Zdroj: autor

U těchto prvků bylo použito několik parametrů:

- `TrackingFunction` je funkce, která proběhne, když je s daným ovládacím prvkem manipulováno. Ve vytvářené CDF aplikaci je použita u proměnných `A` a `P` tak, aby zajistila, že hodnota `P` nepřesáhne hodnotu `A`. Pro uživatele to znamená, že posuvník ovládající proměnnou `P` není možné posunout dále, než na pozici posuvníku `A` a naopak.
- `Enabled` určuje, zda je ovládací prvek aktivní nebo ne. Ve vytvářené aplikaci je ten který prvek aktivní tehdy, když se vypočítává jiná proměnná než ta, kterou prvek ovládá.
- `ControlType` určuje typ ovládacího prvku. V případě standardu, u kterého jsou v CDF aplikaci možné pouze tři volby, stačí jednoduchý přepínač. V případě navýšení počtu standardů lze typ jednoduše změnit.

Zbývá pouze ovládací prvek pro datumy. Mathematica bohužel nemá vestavěný prvek pro volbu datumu. Ve vytvářené aplikaci je toto vyřešeno samostatnými prvky pro koncový a počáteční den, měsíc a rok (kód 3).

Kód 3: Ovládací prvky pro datumy

```
Style[labelt, Bold],
Grid[{
  {"od",
   Control[{{tD1, 1, ""}, Range[1, 31],
     ControlType -> PopupMenu,
     Enabled -> (unknown =!= "t")}],
   Control[{{tM1, 1, ""}, Range[1, 12],
     ControlType -> PopupMenu,
     Enabled -> (unknown =!= "t")}],
   Control[{{tY1, 2000, ""},
     Range[2000, 2100], ControlType -> Slider,
     Enabled -> (unknown =!= "t"),
     Appearance -> "Labeled",
     ImageSize -> Small}]
   },
  {"do",
```

```
  Control[{{tD2, 1, ""}, Range[1, 31],
    ControlType -> PopupMenu,
    Enabled -> (unknown =!= "t")}],
  Control[{{tM2, 1, ""}, Range[1, 12],
    ControlType -> PopupMenu,
    Enabled -> (unknown =!= "t")}],
  Control[{{tY2, 2001, ""}, Range[2000, 2100],
     ControlType -> Slider,
     Enabled -> (unknown =!= "t"),
     Appearance -> "Labeled",
     ImageSize -> Small}]
  }}],
```

Zdroj: autor

Nyní je potřeba naprogramovat výpočetní část aplikace. Nejprve je nutné ošetřit vstupní parametry výpočtu. Proměnné P a A už ošetřené jsou, zbývá ošetřit zadané datumy, tj. aby počáteční datum nepřekročilo koncové datum. Tuto kontrolu by šlo zajistit pomocí `TrackingFunction` u jednotlivých ovládacích prvků (pro dny, měsíce, roky), ale jednodušším řešením je provést kontrolu datumů na začátku bloku `Manipulate`. Tato kontrola probíhá neustále a to tak, že v situaci, kdy uživatel zadá špatně datumy, se koncový datum nastaví na počáteční datum zvýšený o 1 den. Jsou-li datumy správné, vypočítá se podle zvoleného standardu doba úročení. Pro jednotlivé standardy je nutné zavést v inicializační části funkce.

Kód 4: Kontrola vstupu, výpočet doby úročení

```
(* na začátku bloku Manipulate *)
startDate = DateObject[{tY1, tM1, tD1}];
endDate = DateObject[{tY2, tM2, tD2}];

If[startDate >= endDate,
  enableCalculation = False;
  tY2 = tY1;
  tM2 = tM1;
  tD2 = tD1 + 1;
  enableCalculation = True;];

t = tStandard[{tD1, tM1, tY1}, {tD2, tM2, tY2}];

(* u ostatních ovládacích pruků *)
{t, ControlType -> None},
{startDate, ControlType -> None},
{endDate, ControlType -> None},
```

```
(* v inicializační části *)
tStandard30E360[{d1_, m1_, r1_}, {d2_, m2_, r2_}] := (
   360*(r2 - r1) + 30*(m2 - m1) + (d2 - d1))/360 // N;
tStandardACT360[{d1_, m1_, r1_}, {d2_, m2_, r2_}] :=
 DayCount[{r2, m2, d2}, {r1, m1, d1}]/360 // N;
tStandardACT365[{d1_, m1_, r1_}, {d2_, m2_, r2_}] :=
 DayCount[{r2, m2, d2}, {r1, m1, d1}]/365 // N;
```

Zdroj: autor

Dle pravidel pro tvorbu demonstrací musí být proměnné `startDate`, `endDate` a `t` lokální pro daný blok `Manipulate` (*Author Guidelines*, n.d.). Toho lze nejjednodušeji docílit tak, že se pro tyto proměnné zavedou skryté ovládací prvky, které mají nastaven typ ovladače `ControlType -> None`.

Nyní zbývá napsat samotný výpočet neznámých a jeho zobrazení uživateli. Výstup je řešen formou tabulky, přičemž u neznámé proměnné je uveden výsledek výpočtu podle zadaných parametrů. U ostatních proměnných je uvedena jejich hodnota zadaná uživatelem.

Kód 5: Výpočet výsledku a jeho zobrazení

```
If[enableCalculation,
 Grid[{
   {"budoucí hodnota",
    NumberForm[N@If[unknown === "P",
       eP/. Solve[equation/. {eA -> A, ei -> i, et -> t}][[1]], P],
       {Infinity, precision}]},
   {labelA, NumberForm[N@If[unknown === "A",
       eA/. Solve[equation/. {eP -> P, ei -> i, et -> t}][[1]], A],
       {Infinity, precision}]},
   {labeli, NumberForm[N@If[unknown === "i",
       ei/. Solve[equation/. {eP -> P, eA -> A, et -> t}][[1]], i],
       {Infinity, precision}]},
   {labelt <> " (years)", NumberForm[N@If[unknown === "t",
       et/. Solve[equation/. {eP -> P, eA -> A, ei -> i}][[1]], t],
       {Infinity, precision}]}},
   Alignment -> gridAlignment,
   Frame -> gridFrame, ItemSize -> gridSize]]

  (* v inicializační části *)
  equation = eA == eP + eP*ei/100*et;
```

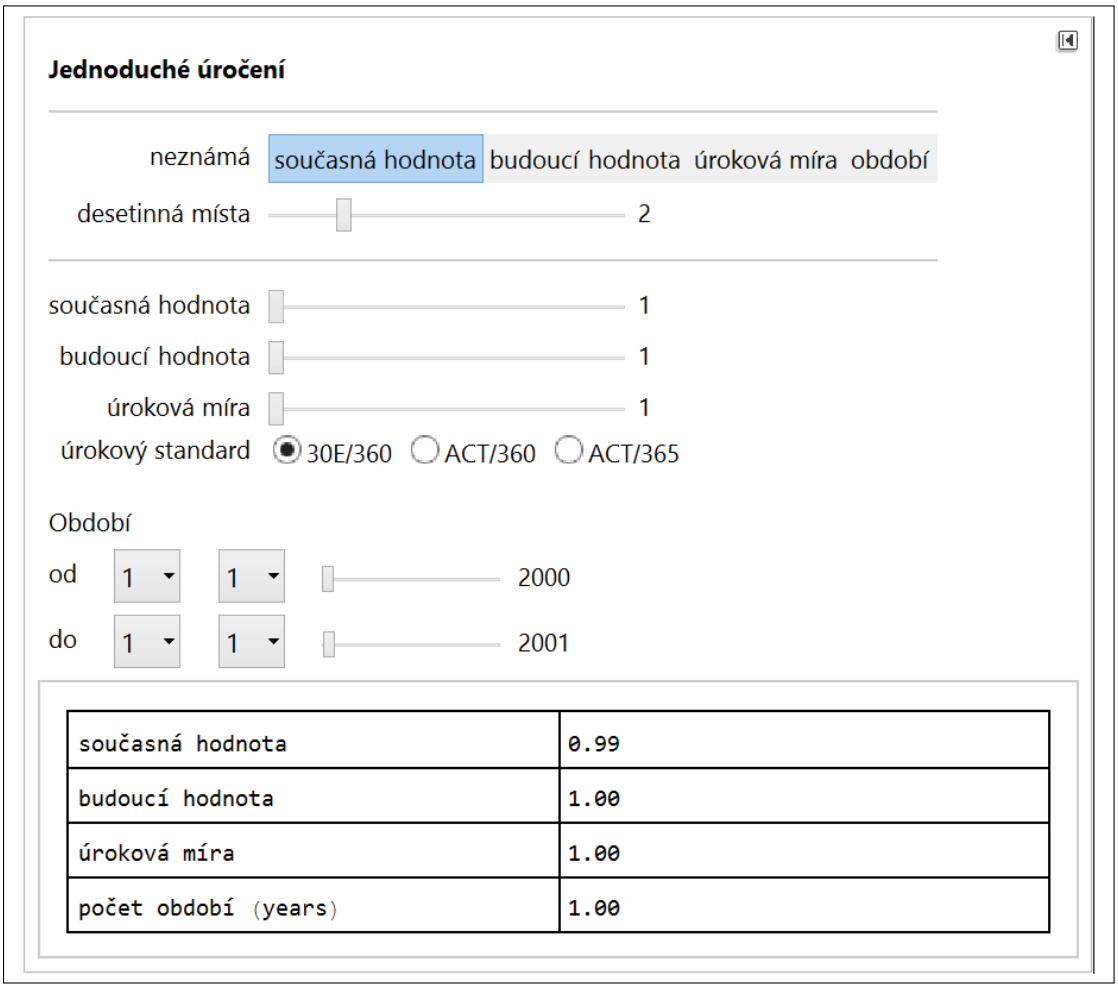
Zdroj: autor

K výpočtu je použita funkce `Solve[]`, která má dva parametry: rovnici a neznámou proměnnou. Do rovnice jsou pomocí operátoru `/.` dosazeny hodnoty známých parametrů.

Výsledek je funkcemi `N[]` a `NumberForm[]` převeden na číslo se zadaným počtem desetinných míst. Jako poslední věc se přidá resetovací tlačítko pomocí příkazu `AppearanceElements -> "ResetButton"` zadaným na úplný konec bloku `Manipulate`. Stisknutím tohoto tlačítka se všechny ovládané proměné v bloku `Manipulate` nastaví na své výchozí hodnoty. Tlačítko je automaticky umístěno do pravého horního rohu.

Na obrázku 5 je výsledná podoba aplikace.

Obrázek 5: Výsledná aplikace pro jednoduché úročení



Zdroj: autor

2.1.2 Složené úročení

Při složeném úročení se úroky přidávají k jistině, takže se počítají „úroky z úroků“.

Vztah pro složené úročení:

$$A = P \cdot \left(1 + \frac{r}{f}\right)^{f \cdot t} \tag{3}$$

kde

- A budoucí hodnota (zúročená částka)
- P základ (kapitál, jistina)
- I úrok
- r roční úroková míra
- f frekvence úročení
- t délka úrokového období (v letech)

V aplikaci se počítá také s efektivní úrokovou mírou. Efektivní úroková míra je taková roční úroková míra, která dává za dobu jednoho roku stejnou splatnou částku jako nominální úroková míra.(Cipra, 2015, str. 40)

Vyjde se z aplikace pro jednoduché úročení, ze které již jsou ovládací prvky současné a budoucí hodnoty, úrokové míry, úrokového standardu a datumů. Přidá se ovládací prvek pro frekvenci úročení:

Kód 6: Ovládací prvek frekvence úročení

```
{{f, 1, "frekvence úročení"}, {
   1 -> "rok",
   (1/2) -> "půlrok",
   (1/4) -> "čtvrtletí",
   (1/12) -> "měsíc"
 },
ControlType -> PopupMenu,
AppearanceElements -> None},
```

Zdroj: autor

Hlavní změna nastane ve výpočetní části aplikace. Výpočet by bylo možné řešit obdobně jako u jednoduchého úročení dosazením známých hodnot do rovnice a nalezení výsledku pomocí `Solve[]`. Jednodušší je však využít vestavěnou funkci `TimeValue[]`. Dle toho, co má být počítáno se do funkce zadá současná nebo budoucí hodnota, efektivní úroková míra a úrokové období (v letech) se záporným nebo kladným znaménkem. K výpočtu

efektivní úrokové míry lze využít vestavěnou funkci `EffectiveInterest[]` s parametry nominální úroková míra (p.a.) a frekvence úročení, např. pro čtvrtletní se zadá 1/4.

Kód 7: Výpočet složeného úročení

```
If[unknown==="P",P=N@TimeValue[A,EffectiveInterest[i/100,f],-t]];
If[unknown==="A",A = N@TimeValue[P,EffectiveInterest[i/100,f],t]];

If[unknown === "i",
  i = 100*(r /. Quiet@FindRoot[
  TimeValue[P, EffectiveInterest[r, f], t] == A, {r, 0}][[1]]);
];
If[unknown === "t",
  t = N[n /. NSolve[
  TimeValue[P,EffectiveInterest[i/100,f], n] == A, n,Reals][[1]]]];
```

Zdroj: autor

K výpočtu nominální úrokové míry je v dokumentaci k funkci `TimeValue[]` doporučeno použit funkci `FindRoot[]`. Té se jako parametr zadá rovnice složeného úročení:

```
TimeValue[P, EffectiveInterest[r, f], t] == A
```

kde neznámou je parametr `r`. Druhým parametrem funkce je právě neznámá a bod, od kterého se má začít hledat. Jelikož úrokové míry nebývají příliš vysoké, je zadán počátek v bodě 0. Samozřejmě je možné zadat i jiný počáteční bod než 0, množina řešení zadané rovnosti zůstává stále stejná. Množina řešení má podobu tzv. pravidel. Pravidlo přiřazuje hodnotu výrazu, například má-li nalezená nominální úroková míra hodnotu 2.45, vypadá množina řešení následovně:

```
{{ r -> 2.45 }}
```

Obecně, existuje-li *n* řešení, má výsledná množina podobu

```
{{ r -> r1 }, {r -> r2}, ----, {r -> rn}}
```

V případě složeného úročení obsahuje množina jeden výsledek (respektive pravidlo). Jedná se tedy o první prvek množiny. K němu se přistoupí operátorem `[[1]]` a poslední krok je dosazení výsledku do proměnné `r` na začátku operátorem `/.` a vynásobení výsledku číslem 100, aby se na výstupu zobrazovala úroková míra rovnou v procentech.

Funkce `FindRoot[]` při svém běhu občas generuje varování ohledně přesnosti výpočtu. Tato varování mohou nastat, jsou-li zadány vstupní parametry (současná a budoucí hodnota, doba úročení) s velmi malými hodnotami v řádu desetin či setin. Tato varování působí rušivě a znepohodlňují užívání aplikace. Aby se varování nezobrazovala, je nutné funkci `FindRoot[]` vložit do bloku `Quiet[]`, ve kterém probíhá vyhodnocení funkce bez zobrazování varovných zpráv.

Výpočet doby úročení je v principu totožný s výše popsaným postupem výpočtu úrokové míry, pouze se namísto `FindRoot[]` používá `NSolve[]`. Není-li uvedeno jinak, `NSolve` počítá na množině komplexních čísel. Vložením parametru `Reals` je nastaveno, aby výpočty probíhaly na množině reálných čísel.

Součástí výstupu aplikace je graf vývoje budoucí hodnoty po jednotlivých rocích. Mathematica podporuje mnoho grafů, pro potřeby vytvářené CDF aplikace se použije `ListPlot[]`. Data jsou do něj zadána v podobě listu. Následující kód slouží k vygenerování listů současné hodnoty a budoucích hodnot v jednotlivých rocích:

Kód 8: Výpočet složeného úročení

```
Avalues =  Table[
  N[TimeValue[P, EffectiveInterest[i/100, f], t1]],
  {t1, 1, Ceiling@t}];

Pvalues = Table[P, Ceiling@t];
chartYMax = TimeValue[P, EffectiveInterest[i/100, f], chartXMax + 1];
```

Zdroj: autor

Funkce `Table[]` generuje list budoucích hodnot od prvního roku do posledního, přičemž poslední rok t je funkcí `Ceiling[]` zaokrouhlen na nejbližší přirozené číslo stejné nebo větší než t. Výsledek t v letech není vždy celé číslo, ale do `Table` nelze zadat interval s nejméně jedním neceločíselným okrajem. Z principu totiž není možné vygenerovat list s například ”jedním a půl”prvky.

Proměnná `chartYMax` slouží k hornímu ohraničení osy y zobrazovaného grafu. Proměnná je nastavena na budoucí hodnotu v roce o 1 větším, než jaký je uživatelem zadaný konečný rok (nechť je označen *Y2*). Toto je učiněno z důvodu lepší viditelnosti sloupce budoucí hodnoty v roce *Y2*. Pokud by byl v grafu zobrazen jako poslední rok *Y2*, popisek hodnot toho roku by nebyl vůbec vidět.

K ovládání grafu jsou použity ovládací prvky pro interval zobrazovaných roků a pro zapínání a vypínání popisků hodnot.

Kód 9: Ovládací prvky grafu

```
{{chartXMin, 1, "od"}, 1, t, 1,
  TrackingFunction -> (
    chartXMin = If[# >= chartXMax, chartXMax - 1, #];&
    ),
  Appearance -> "Labeled",
  AppearanceElements -> None},

{{chartXMax, 2, "do"}, 1, t, 1,
  TrackingFunction -> (
    chartXMax =  If[# <= chartXMin, chartXMin + 1, #];&
    ),
  Appearance -> "Labeled",
  AppearanceElements -> None},

{{chartLabels, False, "popisky"}, {True, False}},
```

Zdroj: autor

V kódu 10 je řešno vykreslení grafu.

Kód 10: Graf složeného úročení

```
ListPlot[
 {Labeled[#, Dynamic@If[chartLabels, #, ""], {Center, Top}] & /@
   Avalues,
  Labeled[#, Dynamic@If[chartLabels, #, ""], {Center, Top}] & /@
   Pvalues},

 PlotRange -> {{chartXMin, chartXMax}, {0, chartYMax}},
 ImageSize -> Medium,
 Filling -> Axis,
 PlotTheme -> {"HeightGrid", "PastelColor"},
 FillingStyle -> Thickness[0.02]
 ]
```

Zdroj: autor

Aby se nad sloupci současné hodnoty zobrazily sloupce budoucí hodnoty, je nutné zadat list s většími hodnotami, tj. list `Avalues` obsahující budoucí hodnoty, jako první prvek množiny listů a list se současnými hodnotami jako druhý. Jinak by se graf nevykresloval správně.

Parametr `PlotRange` určuje rozsah grafu na ose x a y. Pro zobrazení dat jako sloupců, a ne pouze jako bodů, se musí zadat typ vyplnění (`Filling`) pod body grafu. Varianta `Axis` vyplňuje směrem k ose x. Parametr `PlotTheme` určuje celkový vizuální vzhled grafu. `FillingStyle` definuje vzhled výplně pod body, zde je v kódu nastavena poměrně tenká tloušťka, jelikož při zobrazení dat ve velkém časovém intervalu jsou k sobě sloupce přiblíženy více než při zobrazení malých intervalů.

Graf obsahuje popisky dat. Hodnoty popisků se hodnotám přiřazují pomocí tzv. mapování (operátor `/@`) mezi funkcí `Labeled[]` a listem `Avalues`. Mapování funguje tak, že aplikuje první výraz (zde je tím výrazem funkce `Labeled[]`) na každý prvek druhého výrazu (zde je to list `Avalues`). Výsledkem mapování je nový list se stejným počtem prvků jako má list, na který je funkce `Labele[]` mapována. Nový list vypadá takto (níže je uveden pseudokód):

```
{ Labeled[prvek 1, popisek prvku 1,{Center,Top} ],
Labeled[prvek 2, popisek prvku 2,{Center,Top} ],
⋮
Labeled[prvek N, popisek prvku N,{Center,Top} ] }
```

kde N = `Ceiling@t`.

Bohužel se mi nepodařilo přidat do grafu popisky os x a y.

Níže je uveden celý kód pro zobrazování výstupu aplikace:

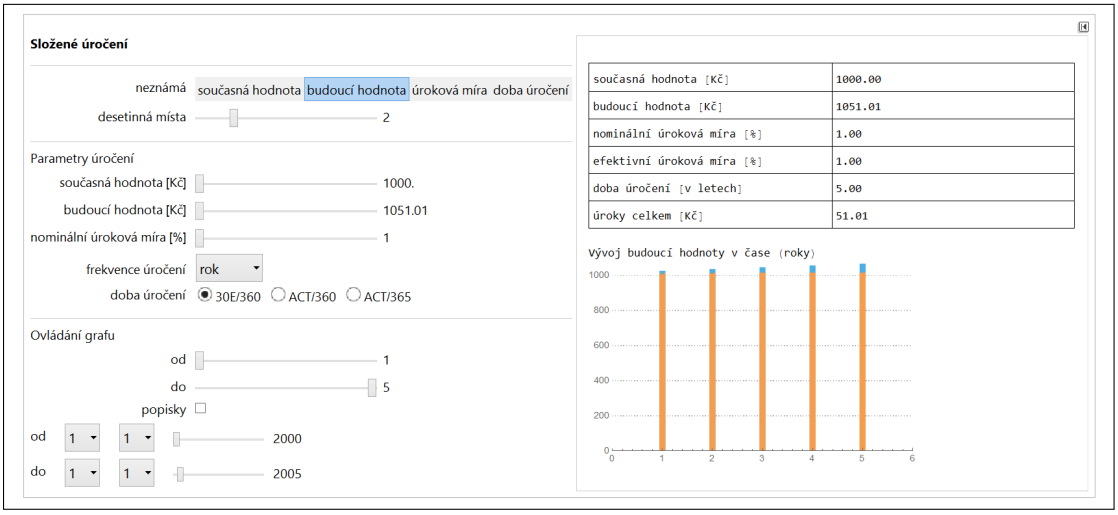
Kód 11: Zobrazování výstupu složeného úročení

```
Column[{Grid[{
    {"současná hodnota [Kč]",NumberForm[N@P,{Infinity,precision}]},
    {"budoucí hodnota [Kč]",NumberForm[N@A,{Infinity,precision}]},
    {"nominální úroková míra [%]",
     NumberForm[N@i, {Infinity, precision}]},
    {"efektivní úroková míra [%]",
     NumberForm[
      N[100*EffectiveInterest[i/100 f], {Infinity,1 + precision}]]},
    {"doba úročení [v letech]",
      NumberForm[N@t, {Infinity, precision}]},
    {"úroky celkem [Kč]", NumberForm[N[A-P],{Infinity, precision}]}
    },
   Alignment -> Left,
   Frame -> All,
   ItemSize -> {20, 2}],
```

```
"",
"Vývoj budoucí hodnoty v čase (roky)",

ListPlot[
 {Labeled[#, Dynamic@If[chartLabels, #, ""], {Center, Top}] & /@
   Avalues,
  Labeled[#, Dynamic@If[chartLabels, #, ""], {Center, Top}] & /@
   Pvalues},
 PlotRange -> {{chartXMin, chartXMax}, {0, chartYMax}},
 ImageSize -> Medium,
 Filling -> Axis,
 AxesLabel -> {"současná hodnota [Kč]", "úrok"},
 PlotTheme -> {"HeightGrid", "PastelColor"},
 FillingStyle -> Thickness[0.02]
 ]
}]
```

Zdroj: autor

Obrázek 6: Výsledná aplikace pro složené úročení



Zdroj: autor

2.2 Základní spořící metody

Spořením rozumíme pravidelné ukládání určité částky po dobu konečné délky. Z hlediska počtu úrokových období se rozeznává spoření krátkodobé, dlouhodobé a kombinované. U krátkodobého spoření se doba spoření rovná právě jednomu úrokovému období, na jehož konci se připíše úrok z úložek. U dlouhodobého spoření se spoří několik úrokových období, úrok z úložky se připisuje na konci období a v dalším období je znovu úročen. Podle toho, kdy je ukládaná částka spořena, se rozlišuje spoření předlhůtní (částka je spořena na začátku úrokového období) a polhůtní (částka spořena na konci období). (Bohanesová, 2006)

2.2.1 Spoření krátkodobé

Pro krátkodobé spoření platí následující vztah.

$$FV = m \cdot K \cdot \left(1 + \frac{m + x}{2m} \cdot i \cdot t\right) \tag{4}$$

kde

- FV naspořená částka
- K úložka
- m počet vkladů za jedno úrokovací období
- i nominální úroková míra (p.a.)
- t doba úročení (v letech)
- x -1 v případě předlhůtního spoření, +1 v případě polhůtního

Vytvářená aplikace bude počítat proměnné FV, K, m a r. Vstup a výstup aplikace je opět řešen standardními ovládacími prvky a mřížkou zobrazující příslušné proměnné (kód 12).

Kód 12: Výstup a výstup krátkodobého spoření

```
(* výstup *)
Column[{
Style[message, Red],
Grid[{
 {"úložka [Kč]", NumberForm[N@K, {Infinity, precision}]},
 {"naspořená částka [Kč]", NumberForm[N@FV, {Infinity, precision}]},
 {"počet úložek [Kč]", NumberForm[N@m, {Infinity, precision}]},
 {"doba úročení [v letech]",
     NumberForm[N[m/12], {Infinity, precision}]},
 {"nominální úroková míra (p.a.) [Kč]",
```

```
     NumberForm[N@i, {Infinity, precision}]}
 },
 Alignment -> Left,
 Frame -> All,
 ItemSize -> gridSize],
  ""
 }],

(* ovládací prvky *)
{{isDue, False, "typ spoření:"}, {
  False -> "předlhůtní",
  True -> "polhůtní"}},
{{K, 1000, "úložka [Kč]"}, 1, 10^6,
 TrackingFunction -> (
   K = If[And[# > FV, unknown =!= "FV"], FV, #]; &),
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "K"),
 AppearanceElements -> None},
{{FV, 10000, "naspořená částka [Kč]"}, 1, 10^6,
 TrackingFunction -> (
   FV = If[And[# < K, unknown =!= "K"], K, #]; &),
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "FV"),
 AppearanceElements -> None
 },
{{i, 0, "nominální úroková míra (p.a.) [\%]"}, 0, 100,
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "i"),
 AppearanceElements -> None
 },
{{m, 1, "počet úložek"}, 1, 12, 1,
 Appearance -> "Labeled",
 Enabled -> (unknown =!= "m"),
 AppearanceElements -> None},
```

Zdroj: autor

Pro výpočetní část aplikace je potřeba definovat vlastní funkci `AnnuityShortTerm` pro krátkodobé spoření (kód 13), jelikož Mathematica obsahuje funkce pouze pro dlouhodobé spoření. Podle parametru `isDue` počítá funkce `AnnuityShortTerm` buď předlhůtní nebo polhůtní spoření.

Kód 13: Výpočetní část aplikace krátkodobého spoření

```
(* ošetření neplatných vstupních hodnot *)
If[m < 0, m = 0];
If[K < 0, K = 0];
```

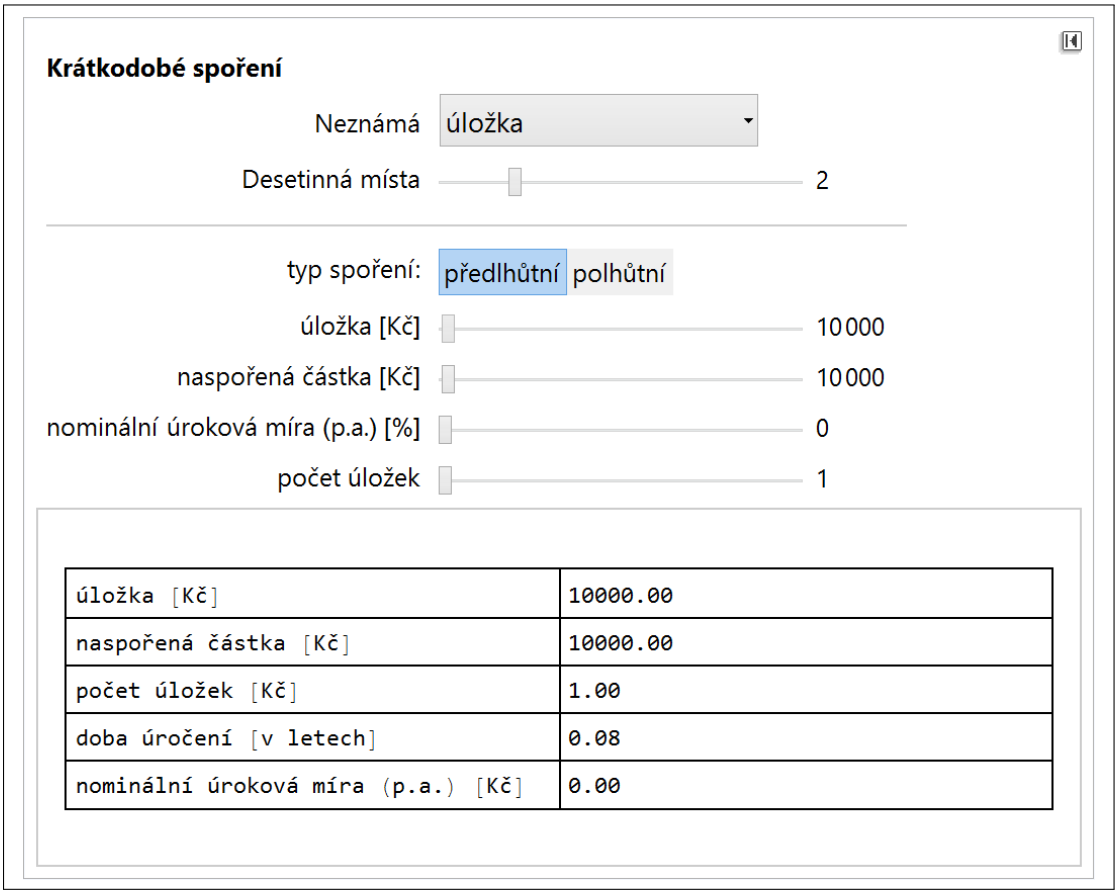
```
If[FV < 0, FV = K];

(* výpočet proměnných *)
message = "";
If[unknown === "m",
  m = Quiet[
   Solve[{AnnuityShortTerm[m1, K, i, isDue] == FV, m1 >= 0}, m1]
   [[All, 1, 2]][[1]]]];

If[unknown === "K",
  K = Quiet[
   Solve[{AnnuityShortTerm[m, K1, i, isDue] == FV, K1 >= 0}, K1]
   [[All, 1, 2]][[1]]]];
If[unknown === "FV",
  FV = Quiet[N@AnnuityShortTerm[m, K, i, isDue]]];

tmpi = Quiet[
  Solve[{AnnuityShortTerm[m, K, i1, isDue] == FV, i1 >= 0},
  i1, Reals]];

If[unknown === "i",
 If[tmpi == {},  i = 0; message = "Nemá řešení",
 i = i1 /. tmpi[[1]]]
];

(* incializační části *)
Initialization :> (
  AnnuityShortTerm[m_, K_, i_, isDue_] :=
    m*K*(1 + i/100*m/12*(m + If[isDue === True, 1, -1])/(2*m));
)
```

Zdroj: autor

Výpočetní část aplikace je jednoduchá - proměnné se vypočítávají kombinací `Solve[]` a `AnnuityShortTerm[]`. U výpočtu úrokové míry se musí ošetřit neřešitelné stavy. Například pro úložku 1 000 Kč, naspořenou částku 10 000 Kč a počet úložek menší nebo rovný 10 je nalezené řešení úrokové míry nenulové nezáporné číslo. Pro počet úložek větší než 10 ale nezáporná úroková míra neexistuje. Množina řešení je uložena do proměnné `tmpi` a v následném bloku `If` se testuje, zda množina je prázdná nebo ne. Je-li prázdná, uloží se do proměnné `message` zpráva o neexistenci řešení. Zpráva se zobrazuje na výstupu (viz začátek kódu 12). Je-li množina neprázdná, uloží se řešení do proměnné `i` a je zobrazeno na výstupu.

Na obrázku 7 je konečná podoba aplikace.

Obrázek 7: Výsledná aplikace pro krátkodobé spoření



Zdroj: autor

2.2.2 Spoření dlouhodobé

Při dlouhodobém spoření je doba spoření delší než úrokové období. Platí vztah

$$FV = K \cdot \frac{(1+i)^k - 1}{i} \cdot x \tag{5}$$

kde

- FV naspořená částka
- K úložka
- f frekvence úročení
- k celkový počet vkladů
- r nominální úroková míra (p.a.)
- i r/f, úroková míra za úrokové období
- x 1 v případě předlhůtního spoření, $(i+1)$ v případě polhůtního

Je-li na spořícím účtu nenulový počáteční stav K_0, zúročí se dle vztahu pro složené úročení

$$K_0 \cdot (1+i)^{nf} \tag{6}$$

a zúročená částka se v rovnici 5 přičte ke zúročeným naspořeným uložkám.

Vytvářená aplikace bude umět vypočítat naspořenou částku, počet plateb, nominální úrokovou míru a výši úložky. Zobrazování výstupu je řešeno tabulkou obdobně, jako u předchozích aplikací. Součástí výstupu je koláčový graf reprezentující strukturu celé naspořené částky, tj. výši úložené částky, úroky z uložené částky, počáteční stav a úroky z počátečního stavu.

Ovládací prvky a zobrazení výsledků v tabulce je z hlediska programového kódu principielně totožné s předchozími aplikacemi. Vztah pro konečnou naspořenou částku lze naprogramovat jako vlastní funkci přesně dle rovnic 5 a 6, ale jednodušší je využít vestavěnou funkci `Annuity` pro předlhůtní spoření a `AnnuityDue` pro polhůtní. Například celková naspořená částka - při úložkách 1 000 Kč, ukládaných na začátku měsíce (předlhůtní spoření) na účet s úrokem 6 % p.a. připisovaným měsíčně po dobu 5 let - se spočítá takto:

```
TimeValue[AnnuityDue[3000, 5*12], 0.06/12, 5*12]
```

Zkombinováním těchto funkcí spolu s funkcí `Solve[]` se lehce vypočtou všechny požadované proměnné. U výpočtu úrokové míry je opět nutné ošetřit neřešitelné stavy. Kód výpočetní části aplikace vypadá následovně:

Kód 14: Výpočet proměnných dlouhodobécho spoření

```
(* výpočet proměnných *)
If[unknown === "K",
  K = N@Quiet[
   Solve[{AnnuityLongTerm[k, K1, i, f, initialDeposit, isDue] == FV,
          K1 >= 0}, K1][[All, 1, 2]][[1]]]];
If[unknown === "FV",
  FV = N@Quiet[AnnuityLongTerm[k, K, i, f, initialDeposit, isDue]]];
If[unknown === "k",
  k = N@Quiet[
       Solve[{AnnuityLongTerm[k1, K, i, f, initialDeposit, isDue]
       == FV, k1 >= 0}, k1][[All, 1, 2]][[1]]]];
If[unknown === "i",
  tmpi = Quiet[
    Solve[{AnnuityLongTerm[k, K, i1, f, initialDeposit,isDue] == FV,
```

```
      i1 >= 0}, i1, Reals]];

If[tmpi == {}, i = 0; message = "Řešení neexistuje",
  i = i1 /. tmpi[[1]] // N;
  message = ""];
];

initialDepositInterest =
  Quiet[AnnuityLongTerm[k, 0, i, f, initialDeposit, isDue]] -
   initialDeposit;
depositsBase = k*K;
depositsInterest =
  Quiet[AnnuityLongTerm[k, K, i, f, 0, isDue]] - depositsBase;

(* funkce pro dlouhodobé spoření
   včetně úročení počátečního stavu *)
Initialization :> (
  AnnuityLongTerm[k_, K_, i_, f_, initialDeposit_, isDue_] := (
   initialDeposit*(1 + i/(100*f))^k +
    If[isDue === True,
      TimeValue[AnnuityDue[K, k], i/(100*f), k],
      TimeValue[Annuity[K, k], i/(100*f), k]]
  );
)
```

Zdroj: autor

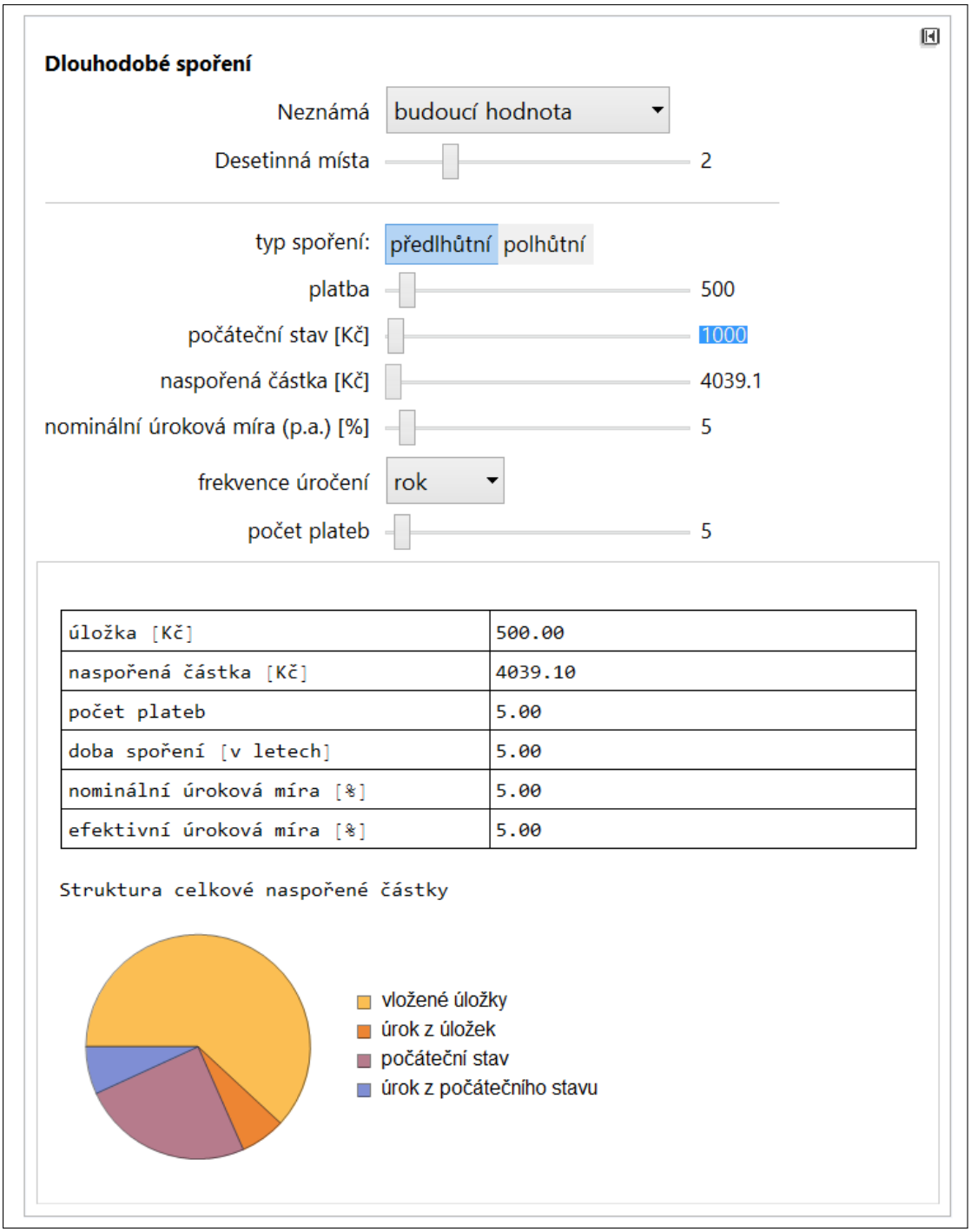
Proměnné `initialDeposit`, `initialDepositInterest`, `depositsBase`, `depositsInterest` obsahují popořadě výši počátečního stavu účtu, celkové úroky z počátečního stavu, celkovou výši úložek uspořených na účtu a celkové úroky z těchto úložek. Následně ve zobrazovací části jsou tyto proměnné vloženy jako seznam do bloku `PieChart[]` (kód 15)

Kód 15: Koláčový graf struktury spoření

```
PieChart[{
  N@depositsBase,
  N@depositsInterest,
  N@initialDeposit,
  N@initialDepositInterest },
 ChartLegends -> {
   "vložené úložky",
   "úrok z úložek",
   "počáteční stav",
   "úrok z počátečního stavu"}]
```

Zdroj: autor

Obrázek 8: Výsledná aplikace pro dlouhodobé spoření



Zdroj: autor

2.2.3 Spoření kombinované

Kombinované spoření se užívá, když se frekvence úročení a frekvence ukládání úložek nerovnají. Platí následující vztah

$$FV = m \cdot K \cdot \left(1 + \frac{m+x}{2m} \cdot r \cdot t\right) \cdot \frac{(1+i)^k - 1}{i} \tag{7}$$

kde

FV naspořená částka

K úložka

f frekvence úročení

m počet vkladů za jedno úrokové období

t úrokové období v letech

k počet plateb

r nominální úroková míra (p.a.)

i r/f, úroková míra za úrokové období

x -1 v případě předlhůtního spoření, +1 v případě polhůtního

Počáteční stav K_0 na účtu se zúročí dle vztahu pro složené úročení (viz rovnice 6).

CDF aplikace pro kombinované spoření se vytvoří upravením aplikace dlouhodobého spoření. V oblasti ovládacích prvků se přidá prvek pro frekvenci vkladů:

Kód 16: Ovládací prvek frekvence plateb

```
 {{fK, 1, "frekvence plateb"},
 {
   1 -> "rok",
   2 -> "půlrok",
   4 -> "čtvrtletí",
   12 -> "měsíc"
 },

  AppearanceElements -> None,
  ControlType -> PopupMenu
}
```

Zdroj: autor

Hlavní změna je ve funkci používané pro výpočet proměnných (kód 17).

Kód 17: Funkce kombinovaného spoření

```
AnnuityGeneral[k_, K_, i_, f_, fK_, initialDeposit_, isDue_] := (
  m = fK/f;
  x = If[isDue == True, 1, -1];

  initialDeposit*(1 + i/(100*f))^k +
  m*K*(1 + (m + x)/(2*m)*i/100*1/f)*
  TimeValue[Annuity[K, k], i/(100*f), k]/K );
```

Zdroj: autor

Proměnná m je určena vztahem

$$m = \frac{\text{frekvence plateb}}{\text{frekvence úročení}}$$

Jelikož funkce `TimeValue[Annuity[]]` odpovídá vztahu:

$$K \cdot \frac{(1+i)^k - 1}{i}$$

ale v rovnici 7 je pouze

$$\frac{(1+i)^k - 1}{i}$$

musí se `TimeValue[Annuity[]]` vydělit proměnnou `K`.

Zbytek kódu aplikace zůstane de facto stejný, jako u dlouhodobého spoření.

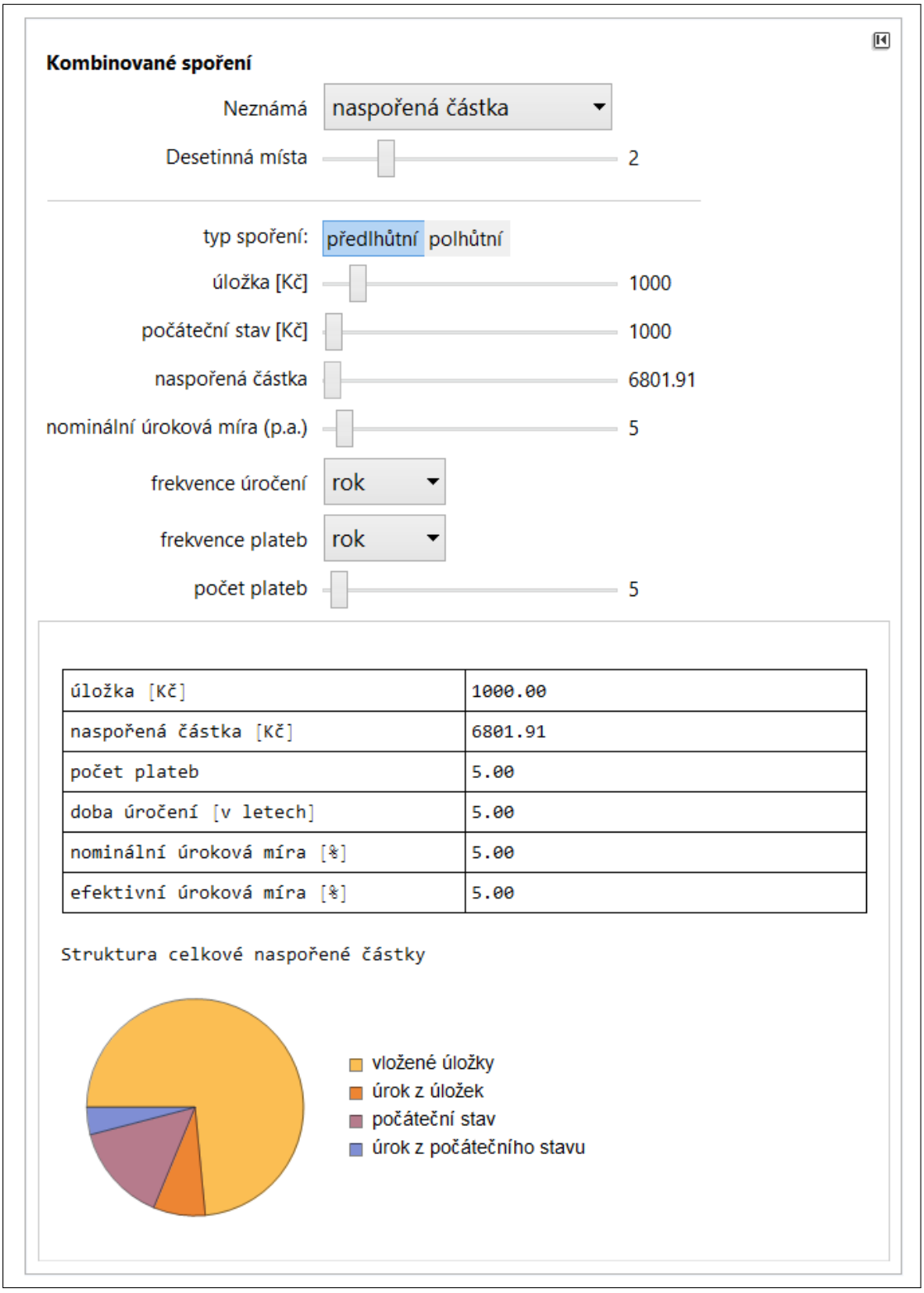
Kód 18: Aplikace kombinovaného spoření

```
(* výpočet proměnných *)
If[unknown === "K", K = N@Quiet[
  Solve[{AnnuityGeneral[k, K1, i, f, fK, initialDeposit, isDue] ==
  FV, K1 >= 0}, K1][[All, 1, 2]][[1]]]];

If[unknown === "FV",
  FV = N@Quiet[AnnuityGeneral[k,K,i,f,fK, initialDeposit, isDue]]];

If[unknown === "k",
  k = N@Quiet[Solve[{
  AnnuityGeneral[k1, K, i, f, fK, initialDeposit, isDue] == FV,
  k1 >= 0}, k1][[All, 1, 2]][[1]]]];
```

```
If[unknown === "i",
  tmpi = Quiet[
    Solve[{AnnuityGeneral[k,K,i1,f,fK,initialDeposit,isDue] == FV,
    i1 >= 0}, i1, Reals]];

If[tmpi == {},
  i = 0;
  message = "Řešení neexistuje",
  i = i1 /. tmpi[[1]] // N;
  message = ""];
];

(* generování dat pro graf *)
initialDepositInterest =
  initialDeposit*(1 + i/(100*f))^k - initialDeposit;
depositsBase = k*K;
depositsInterest =
  Quiet[AnnuityGeneral[k, K, i, f, fK, 0, isDue]] - depositsBase;

(* výstupní tabulka *)
Column[
 {Style[message, Bold, Red],
  Grid[{
   {"úložka [Kč]", NumberForm[N@K, {Infinity, precision}]},
   {"naspořená částka [Kč]",NumberForm[N@FV,{Infinity,precision}]},
   {"počet plateb", NumberForm[N@k, {Infinity, precision}]},
   {"doba úročení [v letech]",
    NumberForm[N[k/f], {Infinity, precision}]},
   {"nominální úroková míra [%]",
    NumberForm[N@i, {Infinity, precision}]},
   {"efektivní úroková míra [%]",
     NumberForm[
     N[100*EffectiveInterest[i/100, 1/f]], {Infinity, precision}]}
  },
  Alignment -> Left,
  Frame -> All,
  ItemSize -> gridSize],
  "",
  (* graf *)
  "Struktura celkové naspořené částky",
  PieChart[{
    N@depositsBase, N@depositsInterest,
    N@initialDeposit,  N@initialDepositInterest},
   ChartLegends -> {
   "vložené úložky",  "úrok z úložek",
   "počáteční stav", "úrok z počátečního stavu"}]}],
```

Zdroj: autor

Obrázek 9: Výsledná aplikace pro kombinované spoření



Zdroj: autor

2.3 Specifické spořící metody

2.3.1 Stavební spoření

Stavební spoření je spoření, kdy peníze jsou ukládány na účet a připisuje se k nim státní podpora ve výši 10 % z vkladu, maximálně však 2 000 Kč. Spoření se uzavírá nejméně na 6 let. Uspořená částka se rovná součtu vkladů, úroků z vkladů a úroků z připsaných státních podpor, snížených o daň z příjmů z těchto úroků a o poplatky účtované stavební spořitelnou.(*Zákon č. 96/1993 Sb.*, 1993)

Nejdříve se princip výpočtu demonstruje níže na příkladu.
Ukládá se 5 000 Kč na konci každého čtvrtletí po dobu 6 let. K 1. dubnu následujícího roku se k naspořené částce připíše příspěvek ve výši 2 000 Kč, příspěvek za 6. rok je připsán na konci 6. roku. Úroková míra je 2 % p.a. s měsíčním připisováním úroků, daněny 15 % daní. Kolik bude po 6 letech naspořeno? (Simerská, 2014)

Řešení sestává ze 2 částí:

- výpočet naspořené částky z pravidelných platbách
- výpočet konečné částky zúročených příspěvků

První část se vypočte dle vztahu pro polhůtní spoření. Nejdříve se určí úroková míra za výplatní období s frekvencí 4.

$$\begin{aligned} \left(1+\frac{i_4}{4}\right)^4 &= \left(1+\frac{i}{12}\right)^{12} \\ \Rightarrow \frac{i_4}{4} &= \left(1+\frac{i}{12}\right)^{12/4} - 1 \\ \frac{i_4}{4} &= \left(1+\frac{0.02\cdot 0.85}{12}\right)^3 - 1 \\ \frac{i_4}{4} &= 0.004256 \end{aligned} \tag{8}$$

Pak se použije vzorec pro budoucí hodnotu polhůtního spoření:

$$FV = K \cdot \frac{(1+i_4/4)^{6*4}-1}{i_4/4} = 5000 \cdot \frac{1.004256^{24}-1}{0.004256} = 126060.8$$

Z úložek se tedy po zúročení a zdanění za 6 let naspoří 126 060 Kč. K této částce se musí přičíst zúročené státní příspěvky. Příspěvek je roven 10 % z ročního vkladu, tedy

$0.1 \cdot 4 \cdot 5000$ Kč $= 2\,000$ Kč. Frekvence úročení je měsíční, ale počet měsíců, kdy se příspěvky úročí, je pro každý příspěvek různý. První příspěvek je uložen na začátku dubna 2. roku a tedy se úročí celkem $(6-2)\cdot 12+9 = 57$ měsíců. Příspěvek připsaný ve 3. roce spoření se úročí $(6-3)\cdot 12+9 = 45$ měsíců, atd. Příspěvek za 6. rok, jež by byl připsaný k 1. dubnu 7. roku, se neúročí a je připsán ke konci 6. roku. Součet všech příspěvků včetně úroků je dán vztahem

$$S = 2000 \cdot \left[(1+i/12)^{57} + \cdots + (1+i/12)^{9} + (1+i/12)^{0}\right] = 12481.28$$

Výsledná naspořená částka celého stavebního spoření je součet

126 060.8 Kč + 12 481.28 Kč = 138 542.07 Kč.

CDF aplikace pro stavební spoření vypočítá naspořenou částku včetně úroků a zdanění, zobrazí strukturu naspořené částky a v tabulce zobrazí pro každý rok spoření:

- výši ročního vkladu
- kumulativní výši vkladů (bez úroků)
- kumulativní úroky z vkladů včetně zdanění
- kumulativní vklady včetně úroků a zdanění
- výši státní podpory za daný rok
- státní podporu za daný rok, zúročenou ke konci spoření
- počet období, po které byla daná podpora úročena

Ovládací prvky aplikace jsou naprogramovány obdobně jako v předchozích aplikacích, kód se principielně nijak neliší od předchozích aplikací.

Stěžejní částí celé aplikace je vlastní funkce `calculateOutput[]`. Funkce vrací několikaúrovňový seznam, který je vložen do bloku `Grid[]` ve zobrazovací části aplikace (kód 19).

Kód 19: Výstupí tabulka stavebního spoření

```
Grid[calculateOutput[ fK*term, K,  (1 - tax/100)*i,  f,  fK,  tax,
  term,  subvention,  subventionMax,  monthSubvention,  isDue,
  feeAccount,  feeAccountFrequency,  feeProduct ],
Alignment -> Right, Frame -> All, ItemSize -> Automatic]
```

Zdroj: autor

Obrázek 10: Tabulka stavebního spoření v CDF aplikaci

Rok N	Roční vklad	Roční vklady (kumulativní)	Úroky z vkladů	Roční vklady vč. úroků	Státní podpora v roce (N+1)	Státní podpora v roce (N+1),zúročená ke konci spoření	Počet úrok. období státní podpory z r. N+1
1	20400	20400	112.575	20512.6	2000.	2117.25	57
2	20400	40800	472.659	41272.7	2000.	2092.01	45
3	20400	61200	1083.24	62283.2	2000.	2067.07	33
4	20400	81600	1947.34	83547.3	2000.	2042.42	21
5	20400	102000	3068.01	105068.	2000.	2018.07	9
6	20400	122400	4448.36	126848.	2000.	2000.	0
Celkem	122400	122400	4448.36	126848.	12000.	12336.8	

Zdroj: autor

Funkce má jako argumenty všechny proměnné, které lze na vstupu nastavit, tj.

- počet úrokových období
- platbu ukládanou na účet stavebního spoření
- úrokovou míru včetně daně
- frekvenci úročení
- frekvenci plateb
- daň
- počet let, na který je stavební spoření uzavřeno
- výši státní podpory (jako procento z ročního vkladu)
- strop státní podpory
- zda se jedná o předlhůtní nebo polhůtní model
- poplatek za vedení účtu
- frekvence poplatku
- poplatek za zřízení produktu

Ve funkci se nepracuje se všemi argumenty, nicméně funkci musí být ze vstupu předány všechny, jelikož `Manipulate[]` spouští jen ty funkce, u nichž dojde ke změně jejich libovolného argumentu.

Nyní budou popsány části funkce `calculateOutput[]` tak, jak na sebe navazují. Nejprve se vytvoří pole o rozměru $[$*počet let spoření + 2 (2 = hlavička a patička tabulky)* $] \times 8$. Velikost tabulky se dynamicky mění dle aktuální hodnoty proměnné `term`, která reprezentuje jak dlouho - v letech - se bude spořit. Pole je uloženo do proměnné `res`, která je definována v bloku `Manipulate[]` jako skrytý ovládací prvek a je tedy dostupná v celém bloku.

Kód 20: Tabulka stavebního spoření - vytvoření pole a vyplnění hlavičky

```
(* vytvoreni pole *)
Clear@res; (* smazani minuleho pole *)
res = Array[{# - 1, 0, 0, 0, 0, 0, 0, 0} &, term + 2];

(* hlavicka *)
res[[1]] = {
   Style["Rok\nN", Bold],
   Style["Roční vklad", Bold],
   Style["Roční vklady\n(kumulativní)", Bold],
   Style["Úroky\nz vkladů", Bold],
   Style["Roční vklady\nvč. úroků", Bold],
   Style["Státní podpora\nv roce (N+1)", Bold],
   Style["Státní podpora\nv roce (N+1),zúročená\nke konci spoření",
    Bold],
   "Počet úrok.\nobdobí státní\npodpory z r. N+1"
   };
```

Zdroj: autor

Poté se projdou všechny prvky pole (buňky tabulky) a provedou se následující příkazy:

1. vyplnění ročních vkladů - jednotlivě za každý rok a kumulativně (první dva příkazy v kódu 21)
2. vyplnění úroků z vkladů a zúročených vkladů (3. a 4. příkaz v kódu 21, je použito kombinované spoření)
3. vypočtení výše státní podpory (5. příkaz v kódu 21)
4. vypočtení kolikrát se má státní podpora z daného roku úročit (6. příkaz v kódu 21) a zapsání těchto hodnot do tabulky (poslední příkaz v kódu 21)
5. zúročení vkladů (předposlední příkaz)

Kód 21: Tabulka stavebního spoření - vyplnění tabulky

```
For[a = 2, a <= term + 1, a++,
  (* rocni vklad (jednotlive, kumulativne) *)
  res[[a]][[2]] = fK*K;
  res[[a]][[3]] = fK*K + If[a <= 2, 0, res[[a - 1]][[3]]];

  (* uroky, vklady vcetne uroku *)
  res[[a]][[5]] = AnnuityGeneral[(a-1)*f,K,i,f,fK,0,isDue] // N;
  res[[a]][[4]] = -res[[a]][[3]] + res[[a]][[5]] // N;
```

```
  (* podpora *)
  res[[a]][[6]] =
   Clip[res[[a]][[2]]*subvention/100, {0, subventionMax}] // N;
  remainingTerms =
   Clip[(term - a)*f + getRemainingTerms[monthSubvention, f], {0,
     Infinity}];
  res[[a]][[7]] = res[[a]][[6]]*(1 + i/(100*f))^remainingTerms;
  res[[a]][[8]] = remainingTerms; ];
```

Zdroj: autor

Následně se vyplní patička tabulky. U sloupců s kumulativními hodnotami se poslední buňka ve sloupci rovná předposlední buňce, u ostatních sloupců se poslední buňka rovná sumě všech předchozích buněk kromě hlavičky.

Kód 22: Tabulka stavebního spoření - patička tabulky

```
(* paticka *)
res[[term + 2]] = {
   Style["Celkem", Bold],
   Style[res[[term + 1]][[3]], Bold],
   Style[res[[term + 1]][[3]], Bold],
   Style[res[[term + 1]][[4]], Bold],
   Style[res[[term + 1]][[5]], Bold],
   Style[Sum[res[[x]][[6]], {x, 2, term + 1}], Bold],
   Style[Sum[res[[x]][[7]], {x, 2, term + 1}], Bold]
   };
```

Zdroj: autor

Dále se vypočítají celkové sumy - suma poplatků, suma zúročených vkladů, suma zúročených státních příspěvků a celková naspořená částka včetně všech úroků a poplatků. Poslední příkaz funkce vrací proměnou `res`, neboli celou tabulku.

Kód 23: Tabulka stavebního spoření - výsledné součty

```
feeTotal = feeAccount*feeAccountFrequency*term + feeProduct;
savings = res[[term + 1]][[5]];
subventionTotal = Sum[res[[x]][[7]], {x, 2, term + 1}];
subventionInterest=subventionTotal-Sum[res[[x]][[6]],{x,2,term+1}];
total = savings + subventionTotal - feeTotal;

res
```

Zdroj: autor

Na výstupu aplikace je suma poplatků a suma naspořených vkladů a všech státních podpor včetně úroků a zdanění. Druhou částí výstupu je graf struktury naspořené částky, který byl popsán v textu výše, a tabulka. Graf je naprogramován obdobně, jako u dlouhodobého a obecného spoření. Hodnoty pro graf se vypočítají ze zadaných vstupů a proměnných vypočtených na konci funkce `calculateOutput[]` (kód 23)

Kód 24: Tabulka stavebního spoření - výstup

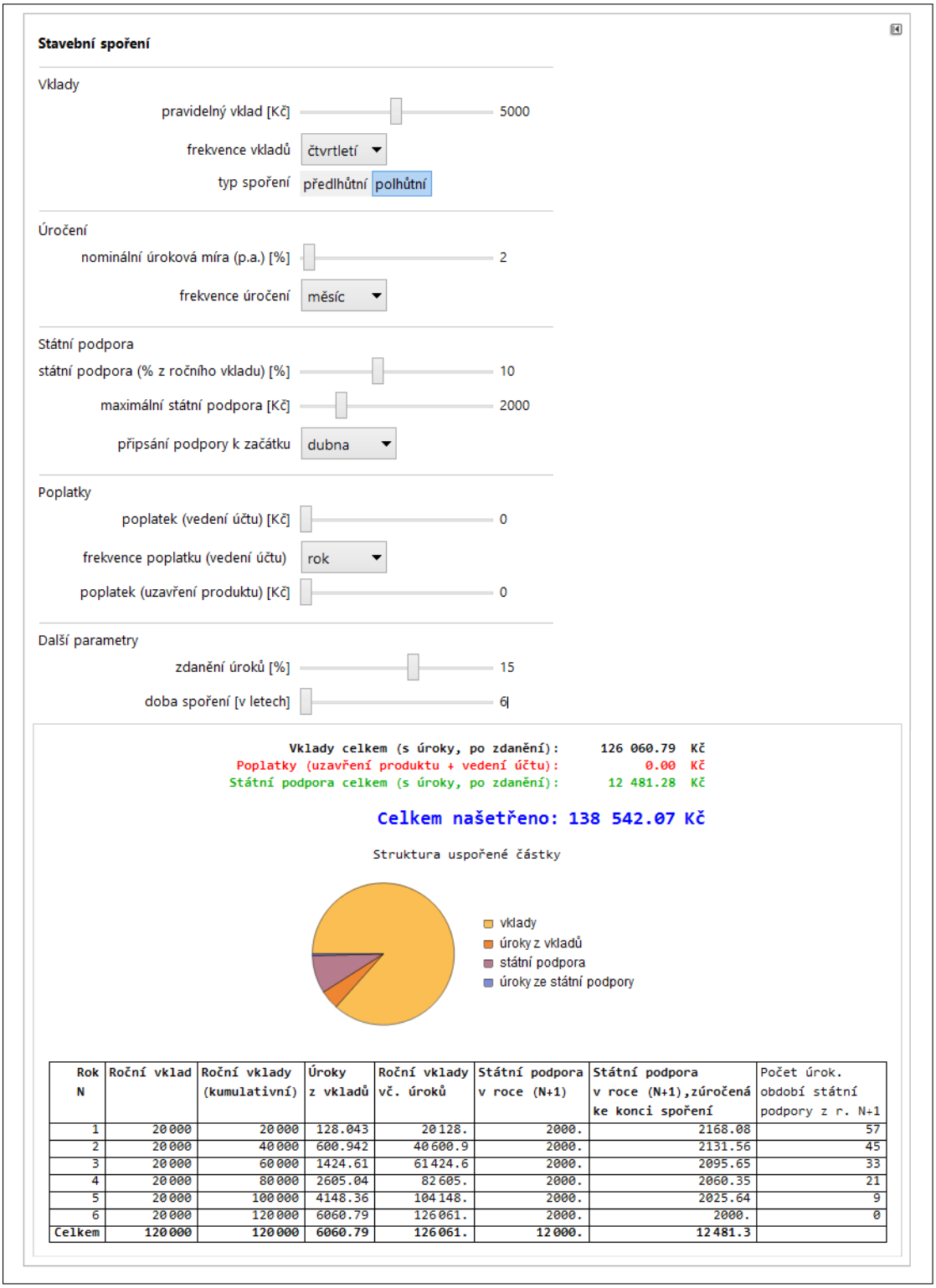
```
Column[{Grid[{
  {Style["Vklady celkem (s úroky, po zdanění):", Bold],
    NumberForm[Style[savings, Bold], {Infinity, 2},
    DigitBlock -> 3,
    NumberSeparator -> " "],
  Style["Kč", Bold]},

  {Style["Poplatky (uzavření produktu + vedení účtu):", Red, Bold],
  Style[ToString@NumberForm[feeTotal, {Infinity, 2},
    DigitBlock -> 3,
    NumberSeparator -> " "],
    Red, Bold],
  Style["Kč", Bold, Red]},

  {Style["Státní podpora celkem (s úroky, po zdanění):",
    Bold, Darker@Green],
    Style[NumberForm[subventionTotal, {Infinity, 2},
      DigitBlock -> 3,
      NumberSeparator -> " "],
      Darker@Green, Bold],
    Style["Kč", Bold, Darker@Green]},
    {},
  {Style["Celkem našetřeno:", Blue, Bold, 20],
   Style[NumberForm[total, {Infinity, 2},
     DigitBlock -> 3,
     NumberSeparator -> " "],
     Blue, Bold, 20],
   Style["Kč", Blue, Bold, 20]}
  },
  Alignment -> Right],
  "",
  "Struktura uspořené částky",
  PieChart[{
    N[fK*term*K],
    N[savings - fK*term*K],
    N[subventionTotal - subventionInterest],
    N[subventionInterest]
    },
```

```
   ChartLegends -> {
     "vklady",
     "úroky z vkladů",
     "státní podpora",
     "úroky ze státní podpory"
     }],
  "",

  Grid[
   calculateOutput[fK*term, K, (1 - tax/100)*i, f, fK, tax, term,
    subvention, subventionMax, monthSubvention, isDue, feeAccount,
    feeAccountFrequency, feeProduct],
   Alignment -> Right,
   Frame -> All,
   ItemSize -> Automatic]
  },
 Alignment -> Center],
```

Zdroj: autor

Ukázka řešení příkladu uvedeného na začátku této podkapitoly s pomocí CDF aplikace je na obrázku 11.

Obrázek 11: Výsledná aplikace pro stavební spoření



Rok N	Roční vklad	Roční vklady (kumulativní)	Úroky z vkladů	Roční vklady vč. úroků	Státní podpora v roce (N+1)	Státní podpora v roce (N+1),zúročená ke konci spoření	Počet úrok. období státní podpory z r. N+1
1	20000	20000	128.043	20128.	2000.	2168.08	57
2	20000	40000	600.942	40600.9	2000.	2131.56	45
3	20000	60000	1424.61	61424.6	2000.	2095.65	33
4	20000	80000	2605.04	82605.	2000.	2060.35	21
5	20000	100000	4148.36	104148.	2000.	2025.64	9
6	20000	120000	6060.79	126061.	2000.	2000.	0
Celkem	120000	120000	6060.79	126061.	12000.	12481.3	

Zdroj: autor

2.3.2 Doplňkové penzijní spoření

Doplňkové penzijní spoření je forma spoření se státní podporou a slouží k zajištění na důchod. Podmínkou pro získání minimálního státního příspěvku je pravidelné ukládání plateb ve výši nejméně 300 Kč měsíčně. Za účastníka doplňkového penzijního spoření může platit platbu nebo její část účastníkův zaměstnavatel, pokud s tím účastník souhlasí.(*Zákon č. 427/2011 Sb.*, 2011) Naspořenou částku lze vyplatit jednorázově - tzv. jednorázové vyrovnání, kdy se výnosy daní 15 % - nebo jako pravidelnou rentu. Jednorázové vyrovnání je dostupné, dosáhl-li účastník věku 60 let a spořil-li alespoň 5 let. (*Doplňkové penzijní spoření (III. pilíř)*, n.d.)

Tabulka 1: Státní příspěvky doplňkového penzijního spoření

Měsíční vklad [Kč]	**Státní příspěvek [Kč]**
100 - 299	0
300 - 399	90 + 20 % z částky nad 300
400 - 499	110 + 20 % z částky nad 400
500 - 599	130 + 20 % z částky nad 500
600 - 699	150 + 20 % z částky nad 600
700 - 799	170 + 20 % z částky nad 700
800 - 899	190 + 20 % z částky nad 800
900 - 999	210 + 20 % z částky nad 900
$\geq$ 1000	230

Zdroj: (*Státní příspěvek penzijního spoření*, n.d.)

Doplňkové penzijní spoření je určeno:

- platbou od účastníka
- platbou od zaměstnavatele
- státním příspěvkem
- dobou spoření
- předpokládaným zhodnocením spoření
- daní z výnosů při jednorázovém vypořádání

Tyto parametry bude možné ovládat v uživatelském rozhraní, výstupem aplikace je na-

spořená částka spolu včetně její struktury a případně i daň.

Na začátku bloku `Manipulate[]` se zjistí, zda na konci spoření přesáhne účastník věkovou hranici pro jednorázové vypořádání. Pokud ano, zpřístupní se ovládací prvek pro výši daně. Pokud ne, daň je nastavena na nulu.

Kód 25: Ošetření jednorázového vypořádání

```
If[age + t >= oneTimePaymentThreshold,
 oneTimePaymentEnabled = True,
 oneTimePaymentEnabled = False;
 oneTimePayment = False;
 tax = 0];

If[oneTimePayment == False, tax = 0];
```

Zdroj: autor

Následně se spočítá spoření funkcí `calculateVariables[]`, které se jako argument zadá účastníkova měsíční platba, měsíční platba zaměstnavatele, zhodnocení spoření, výše daně při jednorázovém vyrovnání a typ spoření (předlhůtní, polhůtní). Funkce spočítá výši vkladů zaměstnavatele i účastníka včetně státní podpory, bez úroků i s úroky, za celé spoření. Výše naspořených částek vkladů se počítá funkcí pro dlouhodobé spoření, která byla popsána v příslušné kapitole.

Kód 26: Výpočet parametrů doplňkového penzijního spoření

```
calculateVariables[K_, Kzam_, i_, t_, tax_, isDue_] := (
 (* zjisteni statniho prispevku *)
 subvention = getSubvention[K];

 (* vklady ucastnika *)
 KTotal = t*(K + subvention);
 KTotalWithInterest = AnnuityLongterm[t, K + subvention, i, isDue];
 KTotalInterest = KTotalWithInterest - KTotal;

 (* vklady zamestnavatele *)
 KzamTotal = Kzam*t;
 KzamTotalWithInterest = AnnuityLongterm[t, Kzam, i, isDue];
 KzamTotalInterest = KzamTotalWithInterest - KzamTotal;

 (* uroky celkem, dan *)
 interestTotal = KTotalInterest + KzamTotalInterest;
 interestTax = interestTotal*tax;
```

```
 (* vysledne castky s uroky, po zdaneni *)
 totalNoInterest = KTotal + KzamTotal;
 totalWithInterest = KzamTotalWithInterest + KTotalWithInterest;
 totalAfterTax = totalWithInterest - interestTax;
);
```

Zdroj: autor

Pro zjištění státního příspěvku dle výše pravidelné účastníkovy platby slouží jednoduchá série podmínek. Dle zákona se státní příspěvky zaokrouhlují na celé koruny dolů, proto se musí užít `Floor[]`.

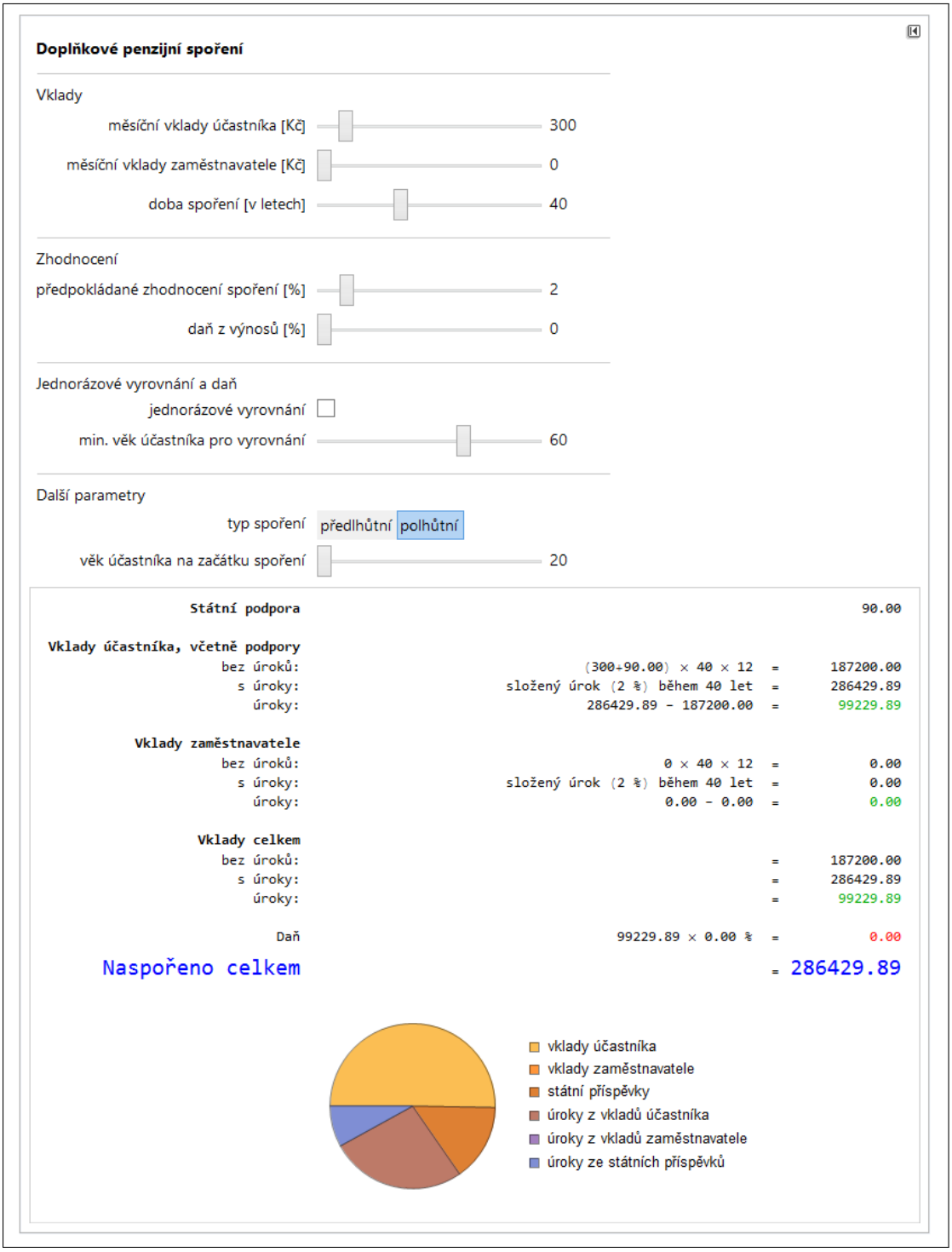
Kód 27: Určení výše příspěvku

```
getSubvention[K_] := (
   sub = 0;
   If[Between[K, {300, 399}], sub = 90 + 0.2*(K - 300)];
   If[Between[K, {400, 499}], sub = 110 + 0.2*(K - 400)];
   If[Between[K, {500, 599}], sub = 130 + 0.2*(K - 500)];
   If[Between[K, {600, 699}], sub = 150 + 0.2*(K - 600)];
   If[Between[K, {700, 799}], sub = 170 + 0.2*(K - 700)];
   If[Between[K, {800, 899}], sub = 190 + 0.2*(K - 800)];
   If[Between[K, {900, 999}], sub = 210 + 0.2*(K - 900)];
   If[K >= 1000, sub = 230];

   Floor@sub
   );
```

Zdroj: autor

Zobrazovací část aplikace má podobu tabulky a koláčového grafu obdobně jako u předchozích aplikací.

Obrázek 12: Výsledná aplikace pro penzijní spoření



Zdroj: autor

2.3.3 Dvoupásmové spoření

U dvoupásmového spoření se zůstatky na spořícím účtu úročí různou úrokovou mírou v závislosti na tom, zda výše zůstatku překročila či nepřekročila hraniční částku. Princip nyní bude ilustrován na příkladu.

Spoří se 3000 Kč měsíčně na účet, kde se zůstatek do 100 000 Kč úročí mírou 1.4 % p.a. měsíčně a zůstatek nad 100 000 Kč mírou 0.8 % p.a. měsíčně. Kolik se naspoří za 5 let?

Řešení: Nejprve je nutné zjistit, za jak dlouho se naspoří 100 000 Kč. Z rovnice dluhodobého spoření se vyjádří počet plateb.

$$\begin{aligned} K \cdot \frac{(1+i)^k - 1}{i} &= FV \\ \implies k &= \frac{ln\,(1 + FV \cdot i/K)}{ln(1+i)} \\ \implies k &= 32.72 \end{aligned}$$

Zůstatek dosáhne částky rovné nebo větší 100 000 Kč po 33 platbách. Naspořená částka po 33 platbách je

$$3000 \cdot \frac{(1 + 0.014/12)^{33} - 1}{0.014/12} = 100870.47$$

Přebytek 870.47 nad hraniční částkou se zúročí mírou 0.8 % ke konci spoření, tj. po dobu $(60 - 33) = 27$ měsíců:

$$870.47 \cdot (1 + 0.008/12)^{27} = 886.27$$

Úrok ze 100 000 Kč je $10000 \cdot 0.014/12 = 116.67$. Tento úrok se přičte k platbě a 27 měsíců bude úročen mírou 0.8 %.

$$(3000 + 116.67) \cdot \frac{(1 + 0.008/12)^{27} - 1}{0.008/12} = 84883.46$$

Celková naspořená částka po 5 letech je 100 000 + 84 883.46 + 886.27 = 185 769.73 Kč.

CDF aplikace vypočítá při kolikáté platbě dojde k překročení hraniční částky, zobrazí úrok z hraniční částky, zúročený přebytek a celkovou naspořenou částku. Zobrazeny budou dva grafy s vývojem zůstatku na účtu během spoření - první graf zobrazí vývoj až do prvního překročení hraniční částky, druhý graf od překročení do konce spoření. V případě, že při dané předhraniční úrokové míře, době spoření a hraniční částce zůstatek nikdy nepřevýší hraniční částku, aplikace na to upozorní a vykreslí se pouze první graf a to jako vývoj zůstatku po celou dobu spoření (jelikož druhý graf je určen pro zůstatky nad hraniční částkou).

Výstupní část je tvořena texty uspořádanými v tabulce, pro ilustraci je v kódu 28 uveden výstup zprávy o nepřekročení hraniční částky (zpráva je v proměnné `message`, která se nastavuje ve výpočetní části aplikace), výstup hraniční částky (vypsaná modře pomocí funkce `Style[]`) a úroku z hraniční částky. `plot1` a `plot2` jsou grafy, rovněž nastaveny ve výpočetní části.

Kód 28: Část textového výstupu dvojpásmového spoření

```
{Style[message, Red, 16]},

{"hraniční částka",
 "", Style[threshold, Blue], "Kč"},

{"úrok z hraniční částky",
 ToString[i] <> " \% p.a. x " <> ToString@threshold,
 thresholdInterest,
 "Kč"},

 (* další výstupy *)
 (* ..... *)
 (* ..... *)

 {plot1, plot2}
```

Zdroj: autor

Ve výpočetní části je použito mnoho vlastních proměnných

- `InterestChangePoint` je číslo platby, při které je překročena hraniční částka
- `InterestChangePointPV` je výše zůstatku v okamžiku překročení hraniční částky
- `InterestChangePointPVOver` je přebytek nad hraniční částkou v okamžiku, kdy ta byla překročena

- `InterestChangePointPVOverFV` je zúročený přebytek
- `thresholdInterest` je úrok z hraniční částky
- `thresholdReached` má hodnotu `True` nebo `False` dle toho, zda při spoření byla překročena hraniční částka
- `afterThresholdFV` je budoucí hodnota úložek včetně úroku z hraniční částky, úročeny jsou nadhraniční úrokovou mírou
- `total` je celková naspořená částka
- `paymentsAfterThreshold` je počet plateb po překročení hraniční částky
- `accountValuesBeforeThreshold` je seznam obsahující zůstatky spořícího účtu až do překročení hraniční částky (v případě, že nebyla překročena, obsahuje zůstatky do konce doby spoření)
- `accountValuesAfterThreshold` je seznam zůstatků po překročení hraniční částky
- `mesageChartBeforeThreshold` je nadpis nad prvním grafem
- `threshold` je hraniční částka

Zjištění, za jak dlouho se naspoří hraniční částka, je provedeno pomocí `Solve[]` s příslušnou rovnici (viz řešení příkladu v úvodu této podkapitoly).

Kód 29: Výpočet překročení hraniční částky

```
getInterestChangePoint[k_, K_, i_, f_, initialDeposit_, isDue_,
   threshold_] := (
   x /. Quiet@
     Solve[{AnnuityLongterm[x, K, i, f, initialDeposit, isDue]
     == threshold}, x, Reals][[1]]
   );
```

Zdroj: autor

Veškeré další výpočty se odehrávají ve vlastní funkci `calculate[]`. Nyní budou kroky v této funkci postupně popsány.

Začne se zjištěním okamžiku dosažení, respektive přesažení, hraniční částky. Je-li počet plateb potřebných k dosažení vyšší než celkový počet plateb během spoření, hraniční částky nebude dosaženo.

Kód 30: Ošetření překročení hraniční částky

```
calculate[k_,K_,i_,i2_,f_,initialDeposit_,isDue_,threshold_]:=(
InterestChangePoint =
 Ceiling@getInterestChangePoint[k, K, i, f, initialDeposit, isDue,
 threshold];

If[InterestChangePoint > k*f,
 thresholdReached = False;
 message = "Hraniční částky nelze\nběhem spoření dosáhnout!",
 thresholdReached = True;
 message = ""];
```

Zdroj: autor

Následně se dle hodnoty proměnné `thresholdReached` nastaví proměnné dle logiky popsané ve výše uvedeném seznamu proměnných a dle početního postupu uvedeného na začátku této podkapitoly.

Kód 31: Nastavení pomocných proměnných dvoupásmového spoření

```
If[thresholdReached == False,
 (* hranicni castka neprekrocena *)
 mesageChartBeforeThreshold = "před dosažením hraniční částky";
 InterestChangePoint = f*k;
 InterestChangePointPV =
 AnnuityLongterm[f*k, K, i, f, initialDeposit, isDue];
 total = InterestChangePointPV;
 InterestChangePointPVOver = 0;
 InterestChangePointPVOverFV = 0;
 thresholdInterest = threshold*i;
 afterThresholdFV = 0;
 paymentsAfterThreshold = 0;
 accountValuesBeforeThreshold =
  Table[AnnuityLongterm[j,K,i,f,initialDeposit,isDue],{j,1,f*k}];
 accountValuesAfterThreshold = {};
 ,

 (* hranicni castka prekrocena *)
 mesageChartBeforeThreshold = "při dosažení hraniční částky";
 InterestChangePointPV =
 AnnuityLongterm[InterestChangePoint,K,i,f,initialDeposit,isDue];
 accountValuesBeforeThreshold =
 Table[AnnuityLongterm[j, K, i, f, initialDeposit, isDue], {j, 1,
    InterestChangePoint}];
```

```
InterestChangePointPVOver = InterestChangePointPV - threshold;
paymentsAfterThreshold = f*k - InterestChangePoint;
InterestChangePointPVOverFV =
 InterestChangePointPVOver*((1 + i2)^(paymentsAfterThreshold));
thresholdInterest = threshold*i;
afterThresholdFV =
 AnnuityLongterm[(f*k - InterestChangePoint), K+thresholdInterest,
  i2, f, initialDeposit, isDue];
```

Zdroj: autor

Pro určení zůstatků po překročení hraniční částky je použita `Table[]`. Položí-li se FV_j rovno výrazu

$$\textit{hraniční částka} + (\textit{přebytek nad hraniční částkou}) \cdot (1+i_2)^j$$
$$+(\textit{ platba} + \textit{úrok z hraniční částky}) \cdot ((1+i_2)^j - 1)/i_2$$

kde i_2 je úroková míra za úrokové období při překročení hraniční částky, jsou jednotlivé zůstatky - v případě, kdy frekvence úročení se rovná frekvenci plateb - postupně

$$FV_j, FV_{j+1}, \cdots, FV_{\textit{počet plateb během celého spoření}}$$

pro j od čísla platby, při které je překročena hraniční částka, do čísla poslední platby, tzn. do konce spoření. Ve funkci `calculate[]` je to s pomocí `Table[]` zapsáno takto:

Kód 32: Generování seznamu zůstatků po překročení hraniční částky

```
accountValuesAfterThreshold = Table[
  threshold
  + InterestChangePointPVOver*(1 + i2)^j
  + AnnuityLongterm[j, K + thresholdInterest, i2, f, initialDeposit,
     isDue],
   {j, 0, f*k - InterestChangePoint}];
```

Zdroj: autor

Dále se uloží celková naspořená částka do proměnné `total` výrazem

```
total = threshold+afterThresholdFV + InterestChangePointPVOverFV;
```

Nyní jsou všechny proměnné již správně nastavené a zbývá už pouze připravit grafy zůstatků na spořícím účtu (kód 33). Kód pro první graf zůstává pořád stejný, akorát je důležité nastavit parametrem `PlotRange` rozsah grafu na ose x (čas, jednotlivé platby)

od 0 do té platby, při níž je poprvé překročena hraniční částka. Rozsah osy y (zůstatek na účtu) je vhodné nastavit od 0 do výše zůstatku na konci spoření (proměnná `total`). Oba dva grafy budou mít na ose y stejné měřítko a bude tudíž možné i pouhým okem srovnat růst zůstatků před a po dosažení hraniční částky, což by při rozdílných měřítcích na osách bylo obtížnější. Hraniční částka je vykreslena červeně pomocí parametrů `GridLines` a `GridLinesStyle`.

Druhý graf se zobrazuje jinak v závislosti na tom, zda byla překročena hraniční částka či nikoliv. Byla-li překročena, nastaví se graf stejně jako první a s příslušným nastavením os. Pokud hraniční částka překročena nebyla, uloží se do `plot2` prázdný graf.

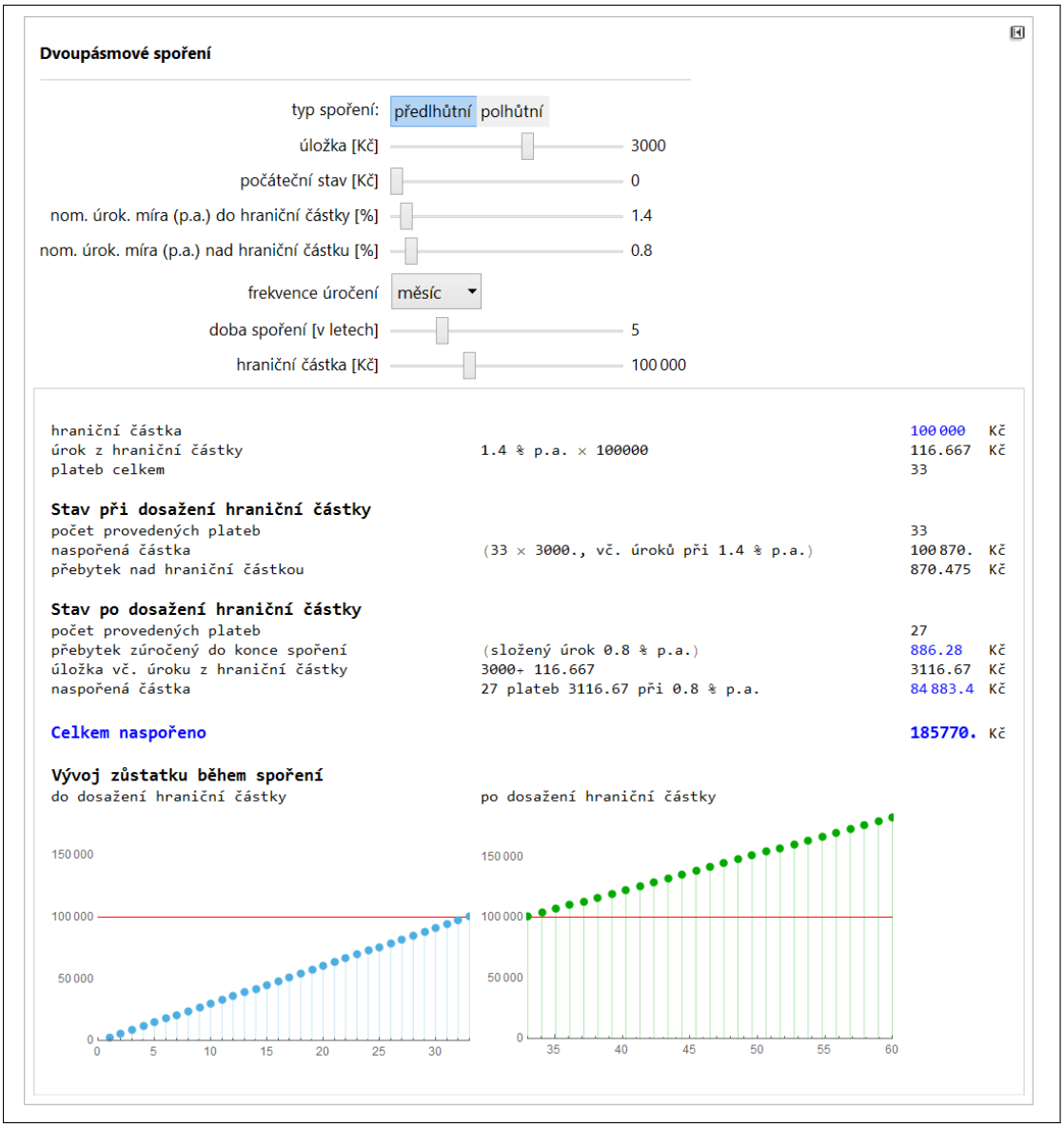
Kód 33: Grafy dvoupásmového spoření

```
plot1 = ListPlot[accountValuesBeforeThreshold,
   Filling -> Axis,
   ImageSize -> Medium,
   PlotStyle -> {PointSize -> Large},
   PlotRange -> {{0, InterestChangePoint}, {0, total}},
   GridLines -> {None, {threshold}},
   GridLinesStyle -> Red,
   PlotTheme -> {"HeightGrid", "PastelColor"}];

If[paymentsAfterThreshold == 0,
  plot2 = ListPlot[{},
    ImageSize -> Medium,
    PlotStyle -> {PointSize -> Large},
    GridLines -> {None, {threshold}},
    PlotRange -> {{f*k, f*k + 10}, {0, total}},
    GridLinesStyle -> Red
    ],
  plot2 = ListPlot[accountValuesAfterThreshold,
    Filling -> Axis,
    ImageSize -> Medium,
    PlotStyle -> {PointSize -> Large, Darker@Green},
    DataRange -> {InterestChangePoint, f*k + 1},
    PlotRange -> {{InterestChangePoint, f*k}, {0, total}},
    GridLines -> {None, {threshold}},
    GridLinesStyle -> Red,
    PlotTheme -> {"HeightGrid", "PastelColor"}
    ]];
```

Zdroj: autor

Na obrázku 13 je pomocí výsledné CDF aplikace řešen příklad z úvodu této podkapitoly.

Obrázek 13: Výsledná aplikace dvoupásmového spoření



Zdroj: autor

3 CDF na internetu

V této kapitole jsou popsány způsoby jak lze CDF dokumenty sdílet na internetu a jak je sdílení realizováno v případě dokumentů z této práce.

Dříve šly dokumenty sdílet přímo na webových stránkách s pomocí CDF pluginu. Tato varianta sdílení však v současnosti není a ani dále nebude možná, jelikož CDF plugin již není podporován v prohlížečích. Společnost Wolfram doporučuje sdílet CDF dokumenty ve Wolfram Cloud nebo zpřístupnit dokumenty na internetu ke stažení a následném otevření ve Wolfram Player.(*Why does Wolfram Research no longer support the CDF plugin?*, n.d.)

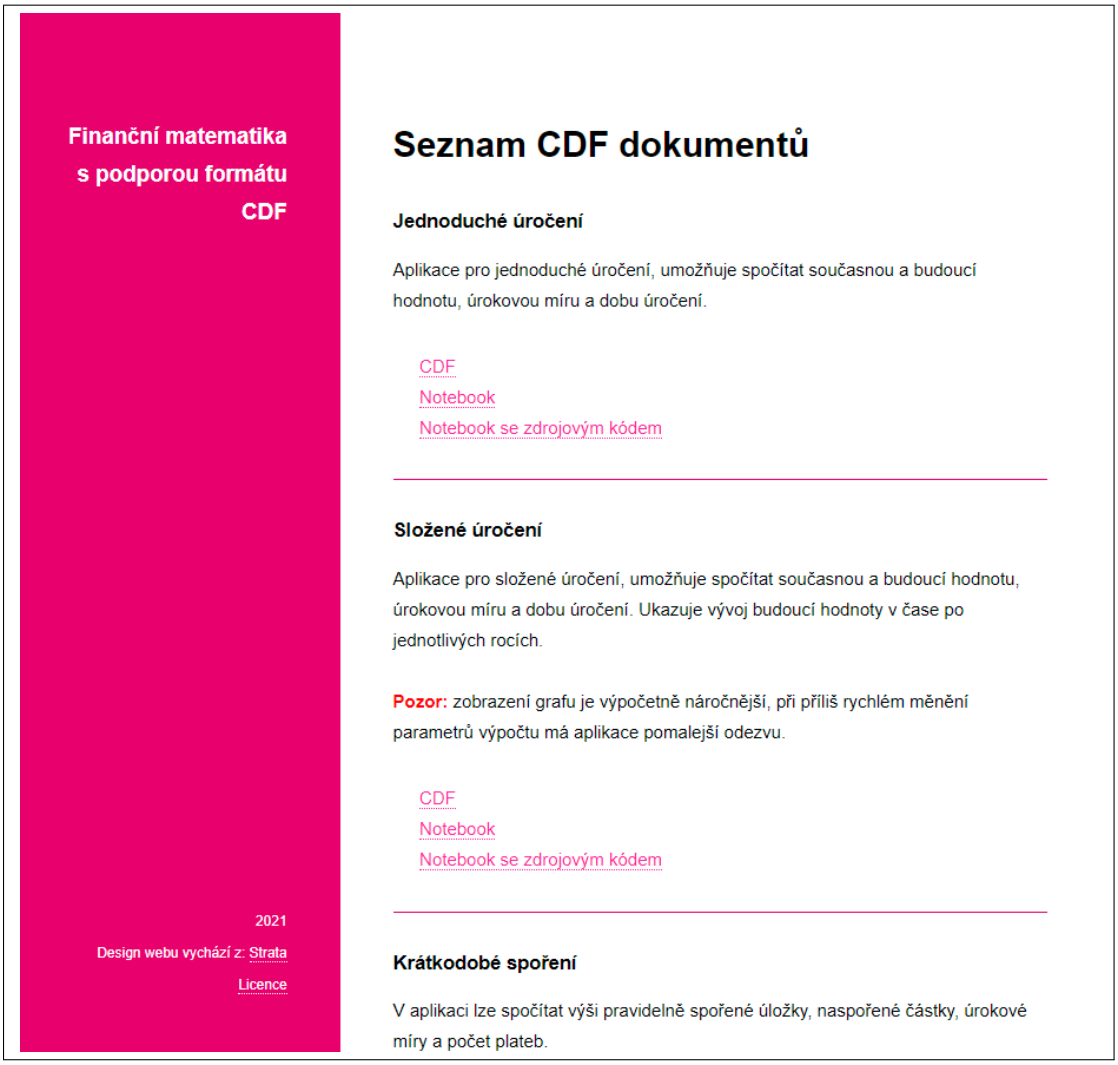
Výhodou Wolfram Cloudu je, že lze soubory jednoduše sdílet přes webový link. Autor může již sdílený soubor upravit, změna se ihned projeví a není potřeba měnit link. Nevýhodami jsou pomalejší odezva aplikací, omezenější možnosti tvorby dokumentů v důsledku některých omezení Wolfram Cloud platformy. To může vést k horší uživatelské přívětivosti daných CDF aplikací. Další nevýhodou je, že v základním plánu všechny soubory uložené v Cloudu po 60 dnech expirují.

Vzhledem k nevýhodám cloudového sdílení dokumentů byly pro zveřejnění CDF dokumentů vytvořených v rámci této práce byly připraveny webové stránky na adrese `http://finmat.ef.jcu.cz`, uložené na serveru fakulty. Stránky jsou vytvořeny tak, aby byly

- přehledné, se snadnou orientací pro uživatele
- přiměřeně informativní
- responzivní, aby se dobře zobrazovaly na různě velikých obrazovkách
- rychlé
- neustále dostupné

Stránky obsahují základní popis CDF, postup zprovoznění CDF dokumentů a návod na jejich užívání. Ke stažení jsou dostupné všechny aplikace jako CDF, notebooky a to včetně zdrojového kódu. Vizuální podoba stránek byla odvozena ze šablony *Strata* dostupné na webu `https://html5up.net`. Šablona je vytvořena standardními webovými technologiemi (HTML, CSS, Javascript) a je sdílená pod licencí Creative Commons CC BY 3.0, dle níž je možné šablonu rozmnožovat, distribuovat a upravovat pro jakýkoliv účel, a to i komerční.(*Creative Commons*, n.d.)

Obrázek 14: Webové stránky s CDF dokumenty



Zdroj: autor

4 Závěr

Cílem této práce bylo vytvořit interaktivní CDF dokumenty pro oblast finanční matematiky a dílčím cílem jejich uveřejnění na internetu.

V teoretické části byly představeny vybrané produkty společnosti Wolfram Research. Byl stručně popsán software *Wolfram Mathematica* a jazyk, ve kterém se v softwaru programuje. Pro tuto práci je v teoretické části nejvíce relevantní zejména popis formátu CDF a jeho využití k tvorbě interaktivních demonstrací.

V praktické části je krok za krokem popsán vývoj jednotlivých aplikací, přičemž důraz je kladen na vysvětlení samotného kódu včetně některých záludností, na které lze v jazyce Wolfram Language narazit. Při vývoji jsem se zpočátku potýkal s problémy vyplývajícími z toho, že Wolfram Language je primárně jazyk zaměřený pro funkcionální a symbolické programování, zatímco já jsem dosud měl zkušenost především s procedurálním stylem programování. U každé aplikace je na začátku kapitoly uvedeno nezbytné teoretické minimum, aby bylo jasné, co se vlastně programuje a čtenář si tak mohl porovnat daný kód aplikace s principem daného problému.

Ve třetí části byly stručně představeny způsoby sdílení CDF dokumentů na internetu. Na serveru fakulty byly vytvořeny webové stránky s uveřejněnými CDF dokumenty spolu s návodem na jejich užívání. Dokumenty si každý může stáhnout a vyzkoušet je.

Možnosti užití formátu CDF jsou široké. Poznatky a informace v této práci lze případně dále využít při tvorbě interaktivních aplikací pro problematiku z jiných kurzů vyučovaných na školách. Osobně by se mi například líbila sada CDF aplikací pro oblast matematické analýzy. Na internetu sice existuje mnoho kalkulátorů a vizualizačních nástrojů, které se tím zabývají, avšak ty se navzájem mohou lišit používaným značením, uživatelským rozhraním a formátem výstupu. Ve Wolfram Mathematica je naproti tomu možné vytvořit sadu aplikací jednotně pokrývajících veškerou látku z daného kurzu.

Summary and keywords

This bachelor thesis focuses on the use of the proprietary interactive Computable Document Format (CDF) by Wolfram Research for financial mathematics. The theoretical part presents selected products of Wolfram Research, describes CDF format together with its most common fields of application.

The practical part deals with the development of CDF applications for selected areas of financial mathematics. It shows the process of developing these applications using the Wolfram Mathematica software.

Partial aim of this thesis is to publish these applications online in order to make them easily accessible to the general public so that they can serve as a supporting material for teaching or learning financial mathematics. For this purpose a website was created on the faculty server. All the developed CDF documents together with their source codes are available for download. The website also contains brief overview of CDFs and tutorial how to use them.

Keywords: financial mathematics, wolfram, cdf, mathematica

Seznam použité literatury

About Wolfram|Alpha. (n.d.). Dostupné z https://www.wolframalpha.com/about/

Author Guidelines. (n.d.). [cit. 21.12.2020]. Dostupné z https://demonstrations.wolfram.com/guidelines.php

Bohanesová, E. (2006). *Finanční matematika I.* Olomouc: Univerzita Palackého. Dostupné z http://oldwww.upol.cz/fileadmin/user_upload/knihovna/Skripta_FF/finan.pdf

Briggs, W. L., Cochran, L. L., & Gillett, B. (n.d.). *MyMathLab® with eText for Calculus, 2/e.* Dostupné z https://www.pearsonhighered.com/briggscochran2einfo/index.html#ebook

Cipra, T. (2015). *Praktický průvodce finanční a pojistnou matematikou* (3rd ed.). Praha: Ekopress.

Creative Commons. (n.d.). Dostupné z https://creativecommons.org/licenses/by/3.0/deed.cs

Doplňkové penzijní spoření (iii. pilíř). (n.d.). [cit. 19.03.2021]. Dostupné z https://www.penize.cz/doplnkove-penzijni-sporeni

Hastings, C. (2015). *An Overview of Mathematica Online.* [online video] [cit. 20.12.2020]. Dostupné z https://www.youtube.com/watch?v=OlkNlLMG_Bg

Maclachlan, F. (2011). *Keynesian Cross Diagram.* Dostupné z https://demonstrations.wolfram.com/KeynesianCrossDiagram/

Notebook Basics. (n.d.). [cit. 18.12.2020]. Dostupné z https://reference.wolfram.com/language/guide/NotebookBasics.html

Shifrin, L. (2008). *Mathematica® programming: an advanced introduction*. Dostupné z http://www.mathprogramming-intro.org/download/MathProgrammingIntro.pdf

Simerská, C. (2014). *Finanční matematika* (1. ed.). Praha: Vysoká škola chemicko-technologická. Dostupné z https://vydavatelstvi.vscht.cz/katalog/publikace?isid=978-80-7080-915-0

Státní příspěvek penzijního spoření. (n.d.). [cit. 19.03.2021]. Dostupné z https://www.finance.cz/duchody-a-davky/penzijni-pripojisteni/abeceda-penzijniho-pripojisteni/statni-prispevek-penzijniho

-pripojisteni/

Torrence, F. B., & Torrence, A. E. (2019). *The Student's Introduction to Mathematica and the Wolfram Language* (3rd ed.). Cambridge, Velká Británie: Cambridge University Press.

Using a Text-Based Interface. (n.d.). [cit. 18.12.2020]. Dostupné z https://reference.wolfram.com/language/tutorial/UsingATextBasedInterface.html

Why does Wolfram Research no longer support the CDF plugin? (n.d.). Dostupné z https://support.wolfram.com/20057

Why Use the Computable Document Format (CDF)? (n.d.). [cit. 20.12.2020]. Dostupné z https://www.wolfram.com/cdf/compare-cdf/

Wolfram, S. (2017). *An Elementary Introduction to the Wolfram Language* (2nd ed.). Wolfram Media, Inc. Dostupné z https://www.wolfram.com/language/elementary-introduction/2nd-ed/index.html

Wolfram Mathematica: The world's definitive system for modern technical computing. (n.d.). [cit. 18.12.2020]. Dostupné z https://www.wolfram.com/mathematica/

Yuan, R. (2014). *Scuderi Split Cycle Engine.* Dostupné z https://demonstrations.wolfram.com/ScuderiSplitCycleEngine/

Zákon č. 427/2011 sb. (2011). *o doplňkovém penzijním spoření*, v aktuálním znění. [cit. 19.03.2021]. Dostupné z https://www.zakonyprolidi.cz/cs/2011-427

Zákon č. 96/1993 sb. (1993). *ze dne 25. února 1993 o stavebním spoření a státní podpoře stavebního spoření*, v aktuálním znění. [cit. 02.03.2021]. Dostupné z https://www.zakonyprolidi.cz/cs/1993-96

Seznam obrázků

Seznam tabulek

Seznam ukázek zdrojových kódů