Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave

DIPLOMOVÁ PRÁCA

Bratislava 2002

Martina Lucová

Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave

Ekonomická a finančná matematika

Numerické Metódy Oceňovania Derivátov Úrokovej Miery

Diplomová práca

Diplomant: Martina Lucová
Vedúci diplomovej práce: doc. RNDr. Daniel Ševčovič, CSc.
Bratislava 2002

Obsah

Úvod

Nevyhnutnou súčasťou kapitálového trhu sú finančné deriváty. Najčastejšie bývajú skúmané dlhopisy, ktoré ponúkajú stabilný príjem a sú najmenej rizikové, takže riziko je takmer nulové. V tejto práci sa zaoberáme najmä bezkupónovými dlhopismi, bez akýchkoľvek transakčných nákladov. V praxi to nie je možné, pretože každý investor sa na kapitálovom trhu stretá s transakčnými nákladmi, najmä vo forme poplatkov za obchodovanie.

V praxi sa často skúma oceňovanie finančných derivátov, hlavne oceňovanie opcií. Oceňovanie iných derivátov nie je veľmi praktizované. Napríklad schémy na oceňovanie dlhopisov nie sú všeobecne známe, preto sme sa rozhodli túto oblasť preskúmať. Cieľom práce bude vymyslieť schému na oceňovanie dlhopisov, podľa ktorej by bolo možné vytvoriť program. Tento program by po zadaní vstupných parametrov počítal hodnoty ľubovoľných finančných derivátov.

V prvej kapitole si priblížime niektoré základné pojmy, ktoré budeme potrebovať pri nasledovnom odvodzovaní. V druhej kapitole sa pozrieme na základné vlastnosti modelov finančných derivátov úrokovej miery. V tretej kapitole si určíme ciele a ich zdôvodnenia. V štvrtej navrhneme numerickú schému na riešenie modelov z druhej kapitoly. V piatej kapitole uvedieme niektoré výsledky numerickej aproximácie hodnôt derivátov a ich porovnania navzájom.

Úprimne ďakujem svojmu vedúcemu diplomovej práce doc. RNDr. Danielovi Ševčovičovi, CSc. za čas, ktorý mi venoval pri príprave, taktiež za odborné a cenné rady, ktoré mi veľmi pomohli pri písaní tejto práce a pri vytváraní programu.

1 Finančné deriváty

Derivát (contingent claim) je cenný papier, ktorého hodnota závisí od nejakých iných obchodovaných cenných papierov, kurzoch cudzích mien, úrokov a iných faktorov.

Finančné deriváty sú nástroje, ktoré sa využívajú na všetkých trhoch (peňažný, devízový, kapitálový aj trh komodít). Vo svete sa s nimi obchoduje už dlhší čas, niečo vyše 20 rokov, na Slovensku bol takýto pokus v roku 1993 na Bratislavskej opčnej burze, ktorá však zanikla. Ich názov pochádza zo slova derivovať, pretože cena je odvodená (derivovaná) od ceny kontrahovaného aktíva. Hlavnou úlohou je ochrana hospodárskych subjektov voči rizikám, vyplývajúcim z budúceho vývoja cien produktov a tovarov, úrokových sadzieb, výmenných kurzov a pod., ale môžu byť zneužité aj na rôzne špekulácie.

1.1 Základné typy derivátov

Základným pojmom na finančnom trhu je forward.

Forward je mimoburzový kontrakt, kde sa jedna strana zaviaže kúpiť alebo predať *underlying instrument*[1] druhej strane v presne stanovenom čase za pevne stanovenú cenu. Kontrakt sa uzatvára medzi dvoma finančnými inštitúciami alebo finančnou inštitúciou a jej klientom. Forwardy sa nepredávajú na burze. Strana, ktorá sa zaviazala akciu kúpiť, je v tzv. *long* pozícii, druhá je v *short* pozícii. Stanovená cena sa nazýva *delivery price.* Cena forwardu je vlastne delivery price, ktorá zabezpečí, aby mal kontrakt na začiatku nulovú cenu pre obe strany. Finančný tok, ktorý vznikne pri ukončení kontraktu v čase T sa nazýva *payoff*, jeho grafom je *payoff diagram.*

Future je vlastne forward obchodovateľný na burze. Vo future kontrakte sa dve strany nepoznajú, preto za daný kontrakt ručí burza. Burza si na ochranu pred neplnením záväzkov jednej alebo oboch strán vytvorí *margin*, čo je vlastne marža(účet), kde každá strana zloží vopred dohodnutú čiastku.

[1] Je nejaké všeobecné aktívum.

Rozdiel medzi forwardom a future je, že future nemá presne stanovený čas nákupu resp. predaja, ale má špecifikované časové obdobie(často je to dlhší časový úsek), v ktorom musí byť kontrakt uzatvorený.

Futurity znamenajú pre obchodníkov príliš veľké riziko (alebo príliš ľahké obohatenie sa) a preto sa čoraz častejšie obchoduje na burzách s opciami.

Opcie sú podmienené termínované obchody, v ktorých sa nejedná o povinnosť niečo predať, resp. kúpiť, ale o právo na predaj, resp. kúpu aktíva, na ktoré je opcia vypísaná. Opcie sa delia na:

Európska call opcia je kontrakt, v ktorom kupujúci získava právo, nie však povinnosť, kúpiť akciu v presne určenom čase (expiračný čas, označme T) za vopred dohodnutú cenu (expiračná cena, označme E). Vypisovateľ opcie je povinný vyhovieť kupujúcemu, pričom za túto službu dostane prémiu (cenu opcie) v čase podpísania kontraktu. Európska call opcia má v expiračnom čase hodnotu závislú od ceny akcie S. Teda ak hodnotu európskej call opcie v čase expirácie označíme $V_{ec}(S,T)$, tak dostávame

$$V_{ec}(S,T) = \max(S - E, 0).$$

Európska put opcia je kontrakt, v ktorom kupujúci získava právo, nie povinnosť, predať akciu v presne určenom čase za vopred dohodnutú cenu. Vypisovateľ opcie je povinný vyhovieť druhej strane, pričom za túto službu dostane prémiu (cenu opcie) v čase podpísania kontraktu. Európska put opcia má v expiračnom čase hodnotu závislú od ceny akcie S, ktorú označíme $V_{ep}(S,T)$,

$$V_{ep}(S,T) = \max(E - S, 0).$$

Americká call(put) opcia sa od európskej líši v tom, že majiteľ opcie má väčšie právo voľby, pretože ju môže uplatniť v ľubovoľnom čase pred expiráciou, zatiaľ čo európsku len v čase expirácie.

Dôležitou úlohou je stanovenie ceny opcie tak, aby pri uzatváraní nebola ani poškodená ani zvýhodnená žiadna zo strán. Ak by to tak nebolo, vznikala by tzv. arbitráž, pri ktorej by sa obohacovala jedna strana na úkor druhej.

1.2 Finančné deriváty úrokovej miery

Deriváty odvodené od úrokovej miery sú v súčasnom svete asi najžiadanejšou skupinou finančných derivátov. Medzi najznámejšie patria put a call opcie, dlhopisy kupónové (coupon bonds) a bezkupónové (zero coupon bonds), Forward contract, Swaps, Caps, Floors, Floating rate dlhopisy a mnohé iné.

V tejto diplomovej práci sa sústredíme na dlhopisy, ktoré vyplácajú fixný príjem a sú najbežnejším typom.

Dlhopis je cenný papier, v ktorom sa dlžník zaväzuje majiteľovi dlhopisu, že v stanovenej lehote splatí nominálnu hodnotu F. Dlhopisy, ktoré okrem nominálnej hodnoty vyplácajú aj pravidelný úrok (kupón (c)) sa nazývajú kupónové dlhopisy. Kupón býva stanovený ako 1% z nominálnej hodnoty dlhopisu, vyplácaný raz alebo dvakrát ročne až do danej lehoty.

Dlhopisy delíme do troch kategórií:

Štátne dlhopisy: sú najvýznamnejšie a najlikvidnejšie dlhopisy. Slúžia na financovanie štátneho rozpočtu a refinancovanie štátneho dlhu. Osobitným typom sú diskontné dlhopisy s ročnou splatnosťou, ktoré sa dostávajú na primárny trh prostredníctvom aukcie a neskôr sa s nimi obchoduje na sekundárnom trhu. Sú kryté vládou a sú považované za bezrizikové. Určujú však trendy na trhu a odvodzujú sa od nich výnosy neštátnych dlhopisov.

Podnikové dlhopisy: sú na trh umiestnené pomocou bánk. Sú menej aktívnejšie ako štátne dlhopisy a aj viac rizikové. Ich výnosy sú rôzne, ktoré sú dané najmä rizikovosťou daného cenného papiera.

Komunálne dlhopisy: slúžia na financovanie dlhodobejších projektov v rámci regiónu, ktoré vydávajú ich miestne správne jednotky, u nás sú to okresy.

Najmenej rizikové sú štátne dlhopisy, a preto sa v ďalšom budeme zaoberať iba týmito dlhopismi.

Hodnotu bezkupónového dlhopisu dostaneme ako

$P = \frac{F}{1+r_n}$, kde r_n závisí od maturity n.

Hodnotu kupónového dlhopisu dostaneme ako

$$P = \sum_{i=1}^{n} \frac{c}{(1+r_i)^i} + \frac{F}{(1+r_n)^n}\,.$$

Cenu bezkupónového dlhopisu s hodnotou 1 v čase t s maturitou T označme ako $P(t,T)$. Budeme pracovať s bezkupónovým dlhopisom, pretože ten je ekvivalentný kupónovému s vhodne dosadenými parametrami.

Spojito úročená diskrétna aktuálna úroková miera v čase t s maturitou T, $R(t,T)$ je definovaná nasledovne:

$$P(t,T) = e^{-R(t,T)(T-t)} \tag{1.4}$$

a teda

$$R(t,T) = -\frac{ln(P(t,T))}{T-t}\,. \tag{1.5}$$

Spojito úročená diskrétna budúca miera v čase t na obdobie $(T, T+\Delta t)$ označená $f(t,T,T+\Delta t)$ je definovaná nasledovne:

$$\frac{P(t,T+\Delta t)}{P(t,T)} = e^{-f(t,T,T+\Delta t)\Delta t}$$

a teda

$$f(t,T,T+\Delta t) = -\frac{ln(P(t,T+\Delta t)) - ln(P(t,T))}{\Delta t}\,.$$

Limity pre $T \to t$, resp. pre $\Delta t \to 0$ definujú okamžitú *short rate* $r(t)$ a okamžitú *forward rate* $f(t,T))$:

$$r(T) = \lim_{T\to t} R(t,T)\,, \tag{1.8}$$

$$f(t,T) = \lim_{\Delta t\to 0}(-\frac{ln(P(t,T+\Delta t)) - ln(P(t,T))}{\Delta t}) = -\frac{\partial \ln(P(t,T))}{\partial T}\,.$$

Vytváraním grafov úrokových mier s dobou splatnosti získame tzv. *term structure interest rates.*

2 Úroková miera

2.1 Deriváty a modely výnosovej krivky

Sú rôzne modely, ktoré možno použiť na oceňovanie opcií. Napríklad také, ktoré predpokladajú, že distribučné rozdelenie úrokovej miery, ceny opcie a iných premenných je lognormálne. Tieto sa však využívajú na oceňovanie derivátov ako tzv. caps[2], Európske opcie pre dlhopisy a Európske opcie pre swapy.[3] Avšak, tieto majú obmedzenia. Nezahŕňajú predpoklad, že by sa úroková miera mohla vyvíjať v čase. Preto sa nemôžu použiť na oceňovanie takých derivátov ako Americké opcie, callable bonds[4] a index amortizing swaps[5].

My však budeme využívať model, ktorí sa snaží prekonať tieto obmedzenia a je známy pod názvom *yield curve model* alebo *term structure model*. Tento model opisuje pravdepodobné správanie všetkých úrokov, ktoré sa menia v čase. Tvar výnosovej krivky sa môže meniť.

Ďalej sa budeme zaoberať rovnovážnymi modelmi a bezarbitrážnymi modelmi. V rovnovážnych modeloch je počiatočná *term structure* ako výstup modelu a v bezarbitrážnych modeloch je počiatočná *term structure* ako vstup do modela.

2.2 Rovnovážne modely

V týchto modeloch sa zvyčajne začína s predpokladmi o ekonomických premenných a odvodzuje sa proces pre krátkodobý bezrizikový úrok r. Tieto skúmajú ako proces vplýva na cenu dlhopisu a cenu opcie. Krátkodobý úrok r, v čase t je úrok aplikovaný na infinitezimálnu krátku časovú periódu. Niekedy sa to nazýva ako okamžitá *short-rate*. Je dôležité

[2] Je to kontrakt, pri ktorom si požičiame tzv. *floating rate* a chceme zabezpečiť nárast úrokových platieb. Cap kontrakt nám zaplatí rozdiel medzi LIBOR a cap úrokom v každom čase.

[3] Je to kontrakt, pri ktorom dochádza k výmene platieb medzi dvoma inštitúciami, kde sa platí fixný úrok k.

[4] Je nástrojom dlhu s uloženými opciami vydanými spoločnosťou. Bond môže byť odkúpený spoločnosťou za určitú cenu a v určitom čase. Držiteľ rozhoduje, či si derivát nechá, alebo ho poskytne spoločnosti.

[5] V tomto kontrakte sa základ pôžičky redukuje na vopred daný. Je skonštrouvaný súhlasne s amorizačným súpisom pôžičky.

zdôrazniť, že v reálnom svete nie je tento proces pre r možný, pretože cena dlhopisu a cena opcie a iné ceny derivátov závisia len na procese zohľadňovanom v rizikovo-neutrálnom svete. Aj tu platia vzťahy, ktoré sú uvedené v predchadzajúcej kapitole ako (1.4) a (1.5). Ukazujú, že ak máme definovaný celý proces pre r, máme aj zadefinovanú počiatočnú *term structure* a aj jej predpokladaný vývoj v budúcnosti.

2.2.1 Jednofaktorové modely

V jednofaktorových modeloch proces pre r, v sebe zahŕňa len jeden zdroj pochybností. Predpokladajme všeobecný vývoj aktuálneho *short-rate* v rizikovo-neutrálnom svete:

$$dr = \mu_r(r,t)dt + \sigma_r(r,t)dw\,. \tag{2.10}$$

Okamžitý drift $\mu_r(r,t)$ a okamžitá štandardná odchýľka $\sigma_r(r,t)$ sú funkciami r, ale nezávislé na čase. Predpoklad jednoduchého faktora nie je obmedzujúci ako sa na prvý pohľad možno zdá. Jednofaktorový model predpokladá, že všetky úroky sa pohybujú rovnakým smerom v krátkodobom intervale, ale už nie v rovnakom počte.

Kedysi predpokladalo, že *term structure* má vždy rovnaký tvar, ale dnes už nie. Práve naopak, v súčasnej dobe má mnoho podôb.

Na odvodenie funkcie ceny bezkupónového dlhopisu potrebujeme Itôovu lemu.

Lema 2.1. *(Itôova lema)*

Nech $f(x,t)$ je hladká funkcia premenných x, t, pričom premenná x je riešením stochastickej diferenciálnej rovnice $dx = \mu(x,t)dt + \sigma(x,t)dw$, kde dw je diferenciál Wienerovho procesu. Potom prvý diferenciál funkcie f je daný vzťahom:

$$df = \sigma(x,t)\frac{\partial f}{\partial x}dw + \left(\frac{\partial f}{\partial t} + \mu(x,t)\frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2(x,t)\frac{\partial^2 f}{\partial x^2}\right)dt.$$

Podľa lemy (2.1) teda odvodíme diferenciál funkcie ceny bezkupónového dlhopisu s dobou splatnosti T:

$$\begin{aligned} dP(t,T) &= \left[\frac{\partial P}{\partial r}\mu_r + \frac{\partial P}{\partial t} + \frac{1}{2}\sigma_r^2\frac{\partial^2 P}{\partial r^2}\right]dt + \sigma_r\frac{\partial P}{\partial r}dw \\ &\equiv \mu dt + \nu dw\,, \end{aligned} \tag{2.12}$$

kde μ resp. ν predstavuje výrazy násobiace dt resp. dw.

Vytvoríme portfólio pozostávajúce z jedného dlhopisu s dobou splatnosti T_1 a θ dlhopisov s dobou splatnosti T_2:

$$\pi(t) = P(t,T_1) + \theta P(t,T_2)\,. \tag{2.13}$$

Zmena ceny tohto portfólia za časový úsek dt je:

$$\begin{aligned} d\pi &= [\mu(t,T_1) + \theta\mu(t,T_2)]dt + \left[\frac{\partial P(t,T_1)}{\partial r}\sigma_r + \theta\frac{\partial P(t,T_2)}{\partial r}\sigma_r\right] dw \\ &\equiv \mu_\mu dt + \sigma_\pi dw\,. \end{aligned} \tag{2.14}$$

Ak položíme

$$\theta = -\frac{\frac{\partial P}{\partial r}(t,T_1)}{\frac{\partial P}{\partial r}(t,T_2)}\,,$$

potom $\sigma_\pi \equiv 0$ a vývoj ceny portfólia je deterministický. Vynásobením čitateľa a menovateľa σ_r dostávame:

$$\theta = -\frac{\nu(t,T_1)}{\nu(t,T_2)}$$

a

$$d\pi = dP(t,T_1) - \frac{\nu(t,T_1)}{\nu(t,T_2)}dP(t,T_2)\,.$$

Keďže portfólio je teraz deterministické, tak platí:

$$d\pi = r(t)\pi(t)dt\,. \tag{2.18}$$

Po dosadení:

$$dP(t,T_1) - \frac{\nu(t,T_1)}{\nu(t,T_2)}dP(t,T_2) = \left[P(t,T_1) - \frac{\nu(t,T_1)}{\nu(t,T_2)}P(t,T_2)\right] r(t)dt\,,$$

$$\mu(t,T_1) - \frac{\nu(t,T_1)}{\nu(t,T_2)}\mu(t,T_2) = \left[P(t,T_1) - \frac{\nu(t,T_1)}{\nu(t,T_2)}P(t,T_2)\right] r(t)\,.$$

Úpravami dostávame :

$$\frac{\mu(t,T_1) - P(t,T_1)r(t)}{\nu(t,T_1)} = \frac{\mu(t,T_2) - P(t,T_2)r(t)}{\nu(t,T_2)}\,. \tag{2.21}$$

Táto rovnica platí pre ľubovoľné doby splatnosti.To znamená, že hodnota zlomkov nezávisí od doby splatnosti, ale je funkciou r a t:

$$\lambda(r,t) = \frac{\mu(t,T) - P(t,T)r(t)}{\nu(t,T)}\,. \tag{2.22}$$

Keby sme absolútny drift a volatilitu vyjadrili percentuálne: $\mu = mP$ a $\nu = \nu' P$, potom:

$$\lambda(t,r) = \frac{m(t,T) - r(t)}{\nu'(t,T)}\,. \tag{2.23}$$

Teda percentálny výnos z dlhopisu sa od bezrizikovej *short-rate* líši o funkciu času a *short-rate* prenásobenej volatilitou. Táto funkcia sa nazýva *market price of risk*. Ak je investorova funkcia konkávna, λ je kladná a reprezentuje výnos navyše oproti bezrizikovému. Tento výnos aj správne kompenzuje riziko z držania dlhopisu.

2.2.2 Všeobecné formy jednofaktorových modelov.

Stochastický proces modelujúci vývoj *short-rate* by mal spĺňať určité kritériá dané historickým vývojom *short-rate* na trhu. Medzi tie najdôležitejšie požiadavky patria:

- Úroky by nemali byť záporné a ani ich hodnota nemôže byť neúnosne vysoká.
- Vysoké hodnoty úroku boli nasledované poklesom a nie rastom, pre nízke úroky to platí naopak. Úrok vykazuje tzv. *mean-reverting.*
- Volatilita úrokov na rôzne obdobia je rôzna, pričom na kratšie obdobia je spravidla vyššia ako na dlhšie.
- Hodnota volatility sa mení s hodnotou *short-rate.*Ukázalo sa, že pre volatilitu v tvare σr^{β} je β najvhodnejšie okolo 1,5.

Samozrejme tieto požiadavky sú obchádzané a dodržiavané sú teda veľmi benevolentne. Žiaden zo známych jednofaktorových modelov nedokáže dodržať tieto body všetky naraz. Preto je dôležité pri aplikácii nejakého modelu dávať pozor, ktoré body najviac ovplyvňujú daný derivát. Obidva modely, ktoré sú spomenuté nižšie, sú v podstate rovnaké. Líšia sa však konkrétnou voľbou parametrov.

2.2.3 Vašíček a Cox - Ingersoll - Ross modely

V roku 1977 navrhol Vašíček stochastický proces pre *short-rate* r vychádzajúc z *tzv.* Orstein-Uhlenbeck procesu v tvare:

$$dr = k(\theta - r)dt + \sigma r^{\beta} dw \equiv: \mu_r dt + \sigma_r dw\,. \tag{2.24}$$

Ďalej cena bezkupónového dlhopisu závisí od r, t, T a vo všeobecnosti spĺňa:

$$\frac{dP}{P} = \mu(r,t,T)dt + \nu(r,t,T)dw\,.$$

Aplikovaním Lemy 2.1 dostaneme:

$$\begin{aligned}
\frac{dP}{P} &= \frac{1}{P}\left[\frac{\partial P}{\partial r}\mu_r + \frac{\partial P}{\partial t} + \frac{1}{P}\frac{\partial^2 P}{\partial r^2}\sigma_r^2\right] dt + \frac{1}{P}\left[\frac{\partial P}{\partial r}\sigma_r\right] dw \\
&= \left[\frac{1}{P}\frac{\partial P}{\partial r}k(\theta - r) + \frac{1}{P}\frac{\partial P}{\partial t} + \frac{1}{2P}\frac{\partial^2 P}{\partial r^2}(\sigma r^{\beta})^2\right] dt + \left[\frac{1}{P}\frac{\partial P}{\partial r}\sigma r^{\beta}\right] dw\,.
\end{aligned}$$

Porovnaním s predchádzajúcim vzťahom zistíme, že:

$$\nu(r,t,T) = \frac{1}{P}\frac{\partial P}{\partial r}\sigma r^{\beta}\,.$$

Z podmienok arbitráže platí:

$$\mu(r,t,T) = r + \lambda^*(r,t)\nu(r,t,T) \Rightarrow \mu(r,t,T) = r + \lambda^*(r,t)\frac{1}{P}\frac{\partial P}{\partial r}\sigma r^{\beta}.$$

Aplikovaním Lemy 2.1 na dP sme dostali jedno vyjadrenie pre drift, z trhovej ceny za riziko pre druhé. Keď ich dáme do rovnosti, dostaneme PDR, ktorú musí spĺňať cena dlhopisu :

$$\frac{\partial P}{\partial r}k(\theta - r) + \frac{\partial P}{\partial t} + \frac{1}{2}\frac{\partial^2 P}{\partial r^2}(\sigma r^{\beta})^2 = rP + \lambda^*(r,t)\frac{\partial P}{\partial r}\sigma r^{\beta}. \tag{2.28}$$

Tu už rozlišujeme modely:

- Vašíčkov model: $\beta = 0,\ \lambda^*(r,t) = \lambda_0$
- CIR model: $\beta = \frac{1}{2},\ \lambda^*(r,t) = \frac{\lambda_0 r^{\frac{1}{2}}}{\sigma}$

pričom λ_0 je konštanta odlišná pre každý z modelov.

Vo Vašíčkovom modeli sa mohlo stať, že úroková miera bola normálne rozdelená a negatívna. V CIR modeli (1985) na odstránenie tohto problému vymysleli alternatívny difúzny proces pre *short-rate* r s inou voľbou β a tým zabezpečili, že úroková miera nebude nikdy negatívna. To znamená, že ak vzrastie krátkodobý úrok, tak vzrastie aj štandardná odchýľka.

Teda môžme zapísať PDR pre jednotlivé modely:

- Vašíček: $rP = \frac{\partial P}{\partial r}[k(\theta - r) - \lambda_0\sigma] + \frac{\partial P}{\partial t} + \frac{1}{2}\frac{\partial^2 P}{\partial r^2}\sigma^2$
- CIR: $rP = \frac{\partial P}{\partial r}(k(\theta - r) - \lambda_0 r) + \frac{\partial P}{\partial t} + \frac{1}{2}\frac{\partial^2 P}{\partial r^2}\sigma^2 r\,.$

Každé riešenie takýchto PDR pri podmienke $P(r,T,T) = 1$ sa dá zapísať vo forme:

$$P(r,t,T) = A(t,T)e^{-B(t,T)r}\,, \tag{2.29}$$

kde $A(t,T)$ a $B(t,T)$ sú konštanty rôzne pre všetky modely.

Tu si ich konkrétne rozvedieme.

- Vašíček:

$$B(t,T) = \frac{1 - e^{-k(T-t)}}{k}$$

$$A(t,T) = exp\left[(B(t,T) - T + t)\left(\theta - \frac{\sigma^2}{2k^2} - \frac{\sigma\lambda_0}{k}\right) - \frac{\sigma^2 B(t,T)^2}{4k}\right]$$

- CIR:

$$B(t,T) = \frac{2(e^{\gamma(T-t)} - 1)}{(k + \lambda + \gamma)(e^{\gamma(T-t)} - 1) + 2\gamma}$$

$$\gamma = \sqrt{(k + \lambda)^2 + 2\sigma^2]}$$

$$A(t,T) = \left[\frac{2\gamma e^{(k+\lambda+\gamma)(T-t)/2}}{(k + \lambda + \gamma)(e^{\gamma(T-t)} - 1) + 2\gamma}\right]^{\frac{2k\theta}{\sigma^2}}$$

V oboch modeloch môže nastať stúpajúci sklon, klesajúci sklon alebo aj nejaká drobná nezrovnalosť. Dlhodobý úrok $R(t,T)$ je lineárne závislý na $r(t)$. To znamená, že hodnota $r(t)$ určuje hladinu *term structure* v čase t. Všeobecný tvar *term structure* v čase t je závislý len na $r(t)$, ale nie na t. CIR model poskytuje formulky pre Európske bezkupónové cally a puty. Európske kupónové opcie môžu byť oceňované pomocou Vašíčkovho modela.

2.3 Bezarbitrážny model

Nevýhoda rovnovážnych modelov je, že nedokážu automaticky fitovať dnešnú *term structure*. Vybraním vhodných parametrov môžu poskytnúť približný fit mnohým *term structure*, s ktorými sa možno stretnúť v praxi. Ale fit nie je presný a v mnohých prípadoch robí značné chyby. Mnohí investori toto považujú za veľmi neuspokojivé. Argumentujú tým, že majú veľmi malú dôveru v cenu dlhopisu, keď model nemá presnú cenu základnej opcie. Aj 1% chyba v cene základného bondu môže znamenať 25% chybu
v opčnej cene.

Bezarbitrážne modely sú skonštruované tak, aby boli presne zhodné s dnešnou *term structure*. Predpokladáme, že *term structure* závisí len od jedného faktora a naznačuje ako môžu byť výsledky zložené z viacerých faktorov.

2.3.1 Hoo and Lee Model

Je to prvý bezarbitrážny model z roku 1986, ktorý bol prezentovaný vo forme binomického stromu. Bol skonštruovaný ako cesta k oceňovaniu množiny dlhopisov bez požiadavky na explicitné preferencie investorovho zisku.

Zohľadňujúci spojité úročenie

$$P(t,T) \equiv \exp(-R(t,T)(T-t))\,,$$

je charakteristický nasledujúcim procesom:

$$dr = \theta(t)dt + \sigma dw\,. \tag{2.36}$$

Má teda dva parametre: štandardnú odchýlku a drift $\theta(t)$, kde r sa hýbe s časom. Term structure určuje závislosť r od času t do splatnosti $T-t$:

$$\theta(t) = \frac{\partial f(0,t)}{\partial t} + \sigma^2 t\,.$$

Použitím Vašíčkovho a CIR modelov dostaneme aj vzťah pre Hoo and Lee model:

$$\frac{\partial P}{\partial t} + \phi(t)\frac{\partial P}{\partial r} + \frac{1}{2}\sigma^2\frac{\partial^2 P}{\partial r^2} - rP\,,$$

kde

$$\phi(t) = \theta(t) - \lambda(t)\sigma$$

a hraničná podmienka je

$$P(r,T,T) = 1\,, \tag{2.40}$$

čo súhlasí s predpokladom, že čistý diskontovaný dlhopis v čase T vyplatí 1.
Ak $\phi(t) = \theta(t)$, tak tento základný predpoklad implikuje, že riskové preferencie investora sú zahrnuté v oceňovaní na trhu čistých dlhopisov, ktoré určujú term structure $\theta(t)$. Riešením modela dostaneme rovnicu

$$P(r,t,T) = A(t,T)e^{-B(t,\tau)r}$$

a konštanty sú rovné:

$$B(t,T) = T-t\,,$$
$$\ln A(t,T) = \ln\frac{P(0,T,r)}{P(0,t,r)} - B(t,T)\frac{\partial \ln P(0,t,r)}{\partial t} - \frac{1}{2}\sigma^2 t B(t,T)^2\,,$$

kde množina $\{P(0,t),\ t \geq 0\}$ je počiatočne exogénne zadaná *Term structure.*

3 Ciele práce

3.1 Nutnosť použitia numeriky

V predchádzajúcej kapitole sme si odvodili explicitné vzorce pre niektoré modely. Vašíčkov a Cox - Ingersoll - Ross model reprezentovali rovnovážne jednofaktorové modely a Ho and Lee model predstavuje bezarbitrážny model.

Keď sa pozrieme na tieto explicitné vzorce, ľahko vidieť, že sú konštruované tak, aby pri riešení spomenutých modelov nevznikali problémy. Tieto vzorce platia iba pre dané tri modely ($\beta = 0, \beta = 1/2, \beta = 1$), pri inej voľbe β už neplatia. Okrajové podmienky spolu s riešením rovnice (2.29) sú totiž umelo vytvorené.

Zamýšľali sme sa ako nájsť okrajové podmienky tak, aby ich spĺňali nielen doteraz známe modely (Vašíčkov, CIR, Ho and Lee), ale aj ďalšie, ktorým dané explicitné riešenie nevyhovuje.

Riešením by mohlo byť použitie numeriky so správnymi okrajovými podmienkami. Dané explicitné riešenia totiž nevyhovujú okrajovým podmienkam.

V prípade, že $r \to \infty,\ B > 0$ pre $t < T,\ \lim\limits_{r\to\infty} Ae^{-Br} = 0$, tak rovnica (2.29) vyhovuje podmienke $P(\infty, t, T) = 0,\ \forall\ 0 < t < T$ a jedna z okrajových podmienok je splnená.

Ako vidíme, rovnica (2.29) nemôže spĺňať druhú podmienku, pretože pre $r \to 0$, $P(0, t, T) = A(t, T) \neq 1$. A to je problém, ktorý sa pokúsime odstrániť.

Z logického hľadiska je na začiatku obchodovania cena dlhopisu maximálna a svoje maximum by nemal prekročiť až do vypršania. A teda správna okrajová podmienka je nasledovná: $P(0, t, T) = 1,\ \forall 0 < t < T$.

V nami navrhnutej schéme na výpočet hodnoty derivátu úrokovej miery budeme vychádzať zo známej okrajovej podmienky a ukážeme, že táto schéma nám dáva výsledky podobné reálnym dátam. Na rozdiel od explicitných vzorcov, ktoré vykazujú rôzne chyby.

4 Numerická analýza oceňovania finančných derivátov

V tejto časti sa budeme venovať numerickej analýze, ktorú potom využijeme pri oceňovaní dlhopisov. Ďalej navrhneme metódu na numerický výpočet hodnoty derivátu. Predpokladáme, že derivát, v našom prípade dlhopis, vyhovuje oceňovacej rovnici (2.28)

4.1 Taylorov polynóm

Definícia 4.1. *Majme $f : \mathcal{O}(a) \to R$, $a \in R$, ktorá je v a n-krát diferencovateľná. Taylorovým polynómom n-tého stupňa z funkcie f v bode a rozumieme:*

$$T_n(f, x, a) = f(a) + \tfrac{f'(a)}{1!}(x - a) + \ldots + \tfrac{f^{(n)}(a)}{n!}(x - a)^n .$$

Zvyškom Taylorovho polynómu rozumieme

$$R_n(f, x, a) =: f(x) - T_n(f, x, a) .$$

Veta 4.1. *Majme $f, g : \mathcal{O}(a) \to R$, $a \in R$, pričom f je v $\mathcal{O}(a)$ (n+1)-krát diferencovateľná, g je v $\mathcal{O}(a)$ diferencovateľná, pričom $g'(x) \neq 0$, $\forall x \in \mathcal{O}(a)$*
$\Rightarrow \forall x \in \mathcal{O}(a)\ \exists \theta \in \langle 0, 1\rangle$

$$R_n(f, x, a) = \tfrac{(x-a)^n}{n!}(1 - \theta)^n \tfrac{g(x)-g(a)}{g'(a+\theta(x-a))} f^{n+1}(a + \theta(x - a))$$

4.2 Numerické riešenie oceňovania finančných derivátov

Na numerické riešenie parciálnej diferenciálnej rovnice použijeme tzv. *metódu sietí.* V oblasti, kde hľadáme riešenie, zvolíme množinu bodov, ktorú nazveme sieťou a príslušné body jej uzlami a nahradíme derivácie príslušnej funkcie, ktoré sa vyskytujú v danej diferenciálnej rovnici a v okrajových podmienkach, diferenciami v uzloch. V tejto metóde sú parciálne derivácie aproximované konečnými diferenciami.

Metóda konečných diferencií je najčastejšie používaná metóda na výpočet hodnoty derivátu, ktorý je zadaný diferenciálnymi rovnicami.

Schému konečných diferencií na oceňovanie opcií aplikovali vo svojich prácach Brennan a Schwartz už v roku 1978, neskôr Courtadon (1982), Geske a Shastri (1985).

Použijeme tzv. *implicitnú metódu* na výpočet hodnoty derivátu P.

Najskôr priestorový interval $\langle 0, R_{max}\rangle$ rozdelíme na n delacich intervalov s krokom

$$h = \frac{R_{max} - 0}{n},$$

pričom deliace body označíme ako $r_i,\ i = 0, 1, ...n$ a $r_i = i\frac{R_{max}}{n}$.

Vo všeobecnosti pre $r \to \infty$ sa už hodnota derivátu vlastne nemení, a preto pre naše účely zvolíme $R_{max} \approx 12$. V našich podmienkach sa táto hranica nemôže zvýšiť a na modelovanie situácie v tejto práci nám postačí.

V parciálnej diferenciálnej rovnici (2.28) urobíme transformáciu

$$\tau = T - t\,, \tag{4.2}$$

kde T je expiračná doba hodnoty derivátu a t je čas. Teda (2.28) sa zmení na

$$\frac{\partial P}{\partial \tau} = k(\theta - r)\frac{\partial P}{\partial r} + \frac{1}{2}(\sigma r^{\beta})^2\frac{\partial^2 P}{\partial r^2} - rP + \lambda\sigma r^{\beta}\frac{\partial P}{\partial r}\,. \tag{4.3}$$

Časový interval $\langle 0, T\rangle$ rozdelíme na m deliacich intervalov s krokom

$$\kappa = \frac{T}{m},$$

s označením deliacich bodov $\tau_j,\ j = 0, 1, ..., m$, čím získame sieť s uzlami

v bodoch $(r_i,\ \tau_j),\ i = 0, 1, ..., n-1,\ j = 0, 1, ..., m$.

Rozviňme funkciu $P(r,\ \tau)$ do Taylorovho polynómu prvého rádu v okolí bodu τ_j, $j = 1, 2, ..., m$ a vypočítame hodnotu $P(r,\ \tau_{j-1})$.

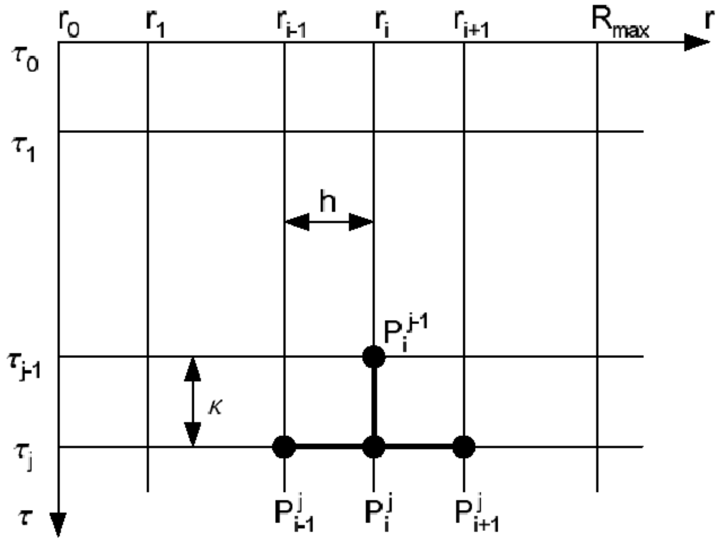
Teda

$$P(r, \tau_j) \approx P(r, \tau_j) + \frac{\partial P(r, \tau_j)}{\partial \tau}(\tau_j - \tau_{j-1}) + O(\tau)\,,$$

kde $\tau = t_j - t_{j-1}$.

Po úprave dostaneme aproximáciu časovej derivácie pomocou spätnej diferencie

$$\frac{\partial P(r, \tau_j)}{\partial \tau} \approx \frac{P(r, \tau_j) - P(r, \tau_{j-1})}{\kappa}\,. \tag{4.6}$$



Obr. 1: Ekvidistantná sieť deliacich bodov

Pre aproximáciu derivácie podľa úroku rozvinieme funkciu $P(r,\tau_j)$ do Taylorovho polynómu druhého rádu v bode r_i, t.j.

$$P(r_{i-1},\tau) = P(r_i,\tau) - \frac{\partial P(r_i,\tau)}{\partial r}(r_i - r_{i-1}) + \frac{1}{2}\frac{\partial^2 P(r_i,\tau)}{\partial r^2}(r_i - r_{i-1})^2 + O(r)\,, \qquad (4.7)$$

$$P(r_{i+1},\tau) = P(r_i,\tau) - \frac{\partial P(r_i,\tau)}{\partial r}(r_{i+1} - r_i) + \frac{1}{2}\frac{\partial^2 P(r_i,\tau)}{\partial r^2}(r_{i+1} - r_i)^2 + O(r)\,, \qquad (4.8)$$

kde $r = |r_{i+1} - r_i|$.

Po odčítaní rovníc (4.7), (4.8) dostaneme aproximáciu prvej derivácie podľa hodnoty úroku v tvare centrálnej diferencie

$$\frac{\partial P(r_i,\tau)}{\partial r} \approx \frac{P(r_{i+1},\tau) - P(r_{i-1},\tau)}{2h}\,. \qquad (4.9)$$

Ak rovnice (4.7), (4.8) sčítame, tak dostaneme aproximáciu druhej derivácie podľa hodnoty úroku

$$\frac{\partial^2 P(r_i,\tau)}{\partial r^2} \approx \frac{P(r_{i-1},\tau) - 2P(r_i,\tau) + P(r_{i+1},\tau)}{h^2}\,. \qquad (4.10)$$

Poznámka: Pre jednoduchosť zápisu budeme označovať hodnotu derivátu $P(r_i,\tau_j)$ ako P_i^j, kde i je priestorový index a j je časový index.

V rovnici (4.3) nahradíme derivácie aproximáciami (4.6), (4.9), (4.10)

$$\begin{aligned} \frac{P_i^j - P_i^{j-1}}{\kappa} &= [k(\theta - r_i) + \lambda\sigma r_i^{\beta}]\frac{P_{i+1}^j - P_{i-1}^j}{2h} \\ &+ \frac{1}{2}\sigma^2 r_i^{2\beta}\frac{P_{i+1}^j - 2P_i^j + P_{i-1}^j}{h^2} - r_i P_i^j\,. \end{aligned}$$

Po úprave dostaneme systém diferenčných rovníc:

$$P_i^{j-1} = a_i P_{i-1}^j + b_i P_i^j + c_i P_{i+1}^j \,, \tag{4.11}$$

kde

$$\begin{aligned} a_i &= \frac{\kappa}{2h} k(\theta - r_i) - \frac{1}{2}\frac{\kappa}{h^2}\sigma^2 r_i^{2\beta} + \frac{\kappa}{2h}\lambda\sigma r_i^{\beta} \,, \\ b_i &= 1 + \frac{\kappa}{h^2}\sigma^2 r_i^{2\beta} + \kappa r_i \,, \\ c_i &= -\frac{\kappa}{2h} k(\theta - r_i) - \frac{1}{2}\frac{\kappa}{h^2}\sigma^2 r_i^{2\beta} - \frac{\kappa}{2h}\lambda\sigma r_i^{2} \end{aligned}$$

a $i = 1, 2, ..., n-1, \; j = 1, 2, ..., m$, kde P^{j-1} sú známe hodnoty a P^j sú neznáme pre danú časovú vrstvu $j = 1, 2, ..., m$.

Hodnotu P^0 poznáme z počiatočnej podmienky a takto dostaneme pre jednu časovú vrstvu j, $n-1$ rovníc o $n-1$ neznámych. Potrebujeme však ešte okrajové podmienky, na nájdenie zvyšných dvoch rovníc. Prvú dostaneme z okrajovej podmienky $P(0, \tau) = 1$, čiže pre $i = 0$ je hodnota P_0 rovná 1, teda

$$P_1^{j-1} - a_1 = b_1 P_1^j + c_1 P_2^j \,. \tag{4.12}$$

Ďalej vieme, že $P(\infty, \tau) = 0$ pre dostatočne veľké R_{max}. Použitím (4.11) a (4.12) dostaneme $n-1$ rovníc o $n-1$ neznámych, pre jednu časovú vrstvu j. Tento systém môžeme napísať do maticového tvaru

$$AP^j = P^{j-1} b \,, \tag{4.13}$$

$j = 1, 2, ..., m$, kde

- $b = (1 - a_1, 1, ..., 1)$ je počiatočný vektor
- $P^j = (P_1^j, ..., P_{n-1}^j)$ je vektor neznámych hodnôt na časovej vrstve j
- $P^{j-1} = (P_1^{j-1}, ..., P_{n-1}^{j-1})$ je vektor známych hodnôt derivátu na časovej vrstve $j-1$
- $A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & & \vdots \\ 0 & a_3 & b_3 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_{n-2} \\ 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{pmatrix}$

je trojdiagonálna matica rozmeru $(n-1) \times (n-1)$.

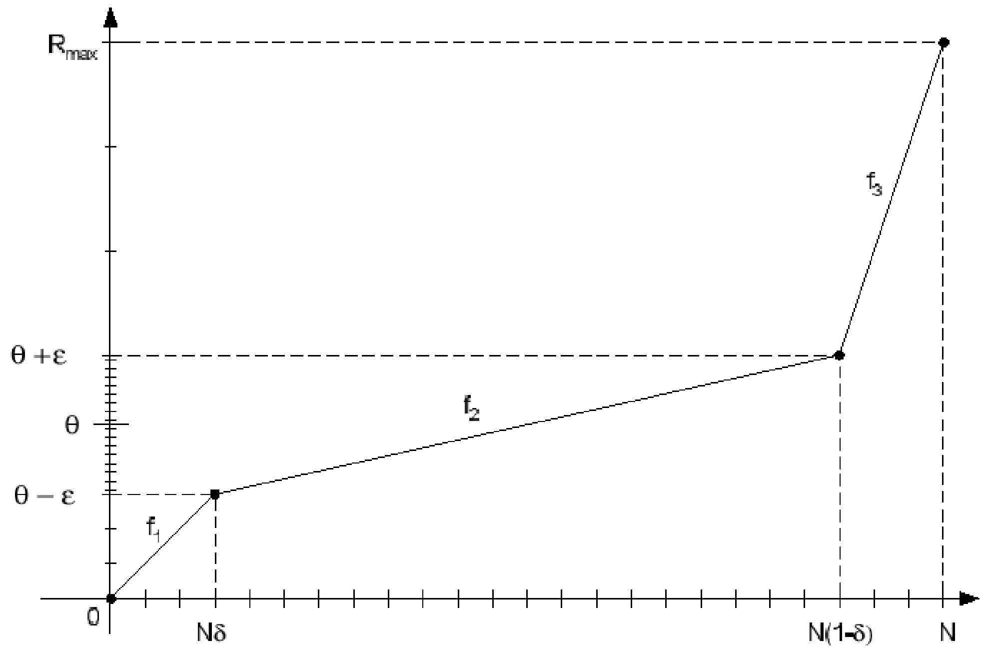
Riešením tohoto systému získame aproximácie hodnôt P na jednotlivých vstvách j a pre jednotlivé hodnoty úroku r.

4.3 Numerické riešenie oceňovania derivátov úrokovej miery

Podobne ako pri numerickom riešení oceňovania finančných derivátov aj v tomto prípade upravíme časový vývoj derivátu substitúciou (4.2) a aproximujeme parciálne derivácie konečnými diferenciami.

Ak by sme spravili ekvidistantné delenie priestorového intervalu, tak pre r blízke R_{max} by to bol prudký pokles smerom dole.

Preto musíme zvoliť vhodnú funkciu, ktorá by nám priestorový interval $\langle 0, R_{max} \rangle$ rozdelila tak, aby počet deliacich bodov v okolí θ bol väčší ako inde. Mohli by sme spraviť aj ekvidistantné delenie, ale potom by výpočet hodnoty derivátu bol veľmi náročný na čas. V rovnomernom delení sme zvolili $r_i = R_{max}\frac{i}{n}$. Nerovnomerné delenie postačí aproximovať zloženou lineárnou funkciou. Na vhodný výpočet budeme potrebovať dva dodatočné parametre ε a δ, pričom ε určuje okolie θ so zvýšeným výskytom deliacich bodov a δ určuje percentuálny podiel množstva priestorových rezov v okolí θ, ktoré je vymedzené parametrom ε.



Obr. 2: Nerovnomerné delenie priestorového intervalu

Vhodnú funkciu sme zložili z troch lineárnych funkciíf_1, f_2, f_3, ktoré sú definované nasledovne:

- $f_1 = i\frac{\theta-\epsilon}{n\delta}$, $i = 1, ..., n-1$, pre $r \in \langle 0,\ \theta - \epsilon)$

- $f_2 = \frac{2\varepsilon}{1-2\delta}[\frac{i}{n} - \delta] + \theta - \epsilon,\ i = 1, ..., n-1$, pre $r \in \langle\theta - \epsilon,\ \theta + \epsilon\rangle$

- $f_3 = \frac{(R_{max} - \theta - \epsilon)}{\delta}\left(\frac{i}{n} - 1 + \delta\right) + \theta + \epsilon,\ i = 1, ..., n-1,$
  pre $r \in (\theta + \epsilon,\ R_{max}\rangle$

Príklad:

Pre $n = 50$; $\delta = 0.05$; $\epsilon = 0.01$; $R_{max} = 12$; $\theta = 0.06$; to znamená, že v okolí θ bude v intervale $\langle\theta - \epsilon,\ \theta + \epsilon\rangle$ 90% všetkých deliacich krokov, čiže v rozmedzí $\langle 0.05, 0.07\rangle$ bude 45 deliacich bodov.

Aproximovaním parciálnych derivácií konečnými diferenciami použitím Taylorovho rozvoja dostávame:

$$\frac{\partial P(r_i, \tau)}{\partial r} \approx \frac{P(r_{i+1}, \tau) - P(r_{i-1}, \tau)}{r_{i+1} - r_{i-1}}\,, \tag{4.14}$$

$$\begin{aligned} \frac{\partial^2 P(r_i, \tau)}{\partial r^2} &\approx \frac{2}{(r_{i+1} - r_{i-1})(r_i - r_{i-1})} P(r_{i-1}) \\ &- \frac{2}{(r_{i+1} - r_i)(r_i - r_{i-1})} P(r_i) \\ &+ \frac{2}{(r_{i+1} - r_{i-1})(r_{i+1} - r_i)} P(r_{i+1})\,. \end{aligned} \tag{4.15}$$

Aproximácia časovej derivácie sa rovná:

$$\frac{\partial P(r, \tau_j)}{\partial \tau} \approx \frac{P(r, \tau_j) - P(r, \tau_{j-1})}{\kappa}\,. \tag{4.16}$$

Keď rovnice (4.14), (4.15), (4.16) dosadíme do rovnice (2.28) dostaneme:

$$\begin{aligned} \frac{P_i^j - P_i^{j-1}}{\kappa} &= [k(\theta - r_i) + \lambda\sigma r_i^{\beta}] \frac{P_{i+1}^j - P_{i-1}^j}{r_{i+1} - r_{i-1}} \\ &+ \sigma^2 r_i^{2\beta} \left[\frac{P_{i-1}^j}{(r_{i+1} - r_{i-1})(r_i - r_{i-1})} - \frac{P_i^j}{(r_{i+1} - r_i)(r_i - r_{i-1})} + \frac{P_{i-1}^j}{(r_{i+1} - r_{i-1})(r_{i+1} - r_i)} \right] \\ &- r_i P_i^j\,. \end{aligned} \tag{4.17}$$

Po úprave

$$P_i^{j-1} = a_i P_{i-1}^j + b_i P_i^j + c_i P_{i+1}^j\,, \tag{4.18}$$

kde

$$
\begin{aligned}
a_i &= \frac{\kappa}{r_{i+1}-r_{i-1}}(k(\theta-r_i)+\lambda\sigma r_i^{\beta})-\frac{\kappa}{(r_{i+1}-r_{i-1})(r_i-r_{i-1})}\sigma^2 r_i^{2\beta}\,,\\
b_i &= 1+\frac{\kappa}{(r_{i+1}-r_i)(r_i-r_{i-1})}\sigma^2 r_i^{2\beta}+\kappa r_i\,,\\
c_i &= -\frac{\kappa}{r_{i+1}-r_{i-1}}(k(\theta-r_i)+\lambda\sigma r_i^{\beta})-\frac{\kappa}{(r_{i+1}-r_{i-1})(r_{i+1}-r_i)}\sigma^2 r_i^{2\beta}
\end{aligned}
$$

a $i=1,2,...,n-1,\ j=1,2,...,m$, kde P^{j-1} sú známe hodnoty a P^j sú neznáme pre danú časovú vrstvu $j=1,2,...,m$.

Hodnotu P^0, podobne ako v predchádzajúcej kapitole, dostaneme z počiatočnej podmienky a tým dostaneme pre jednu časovú vrstvu j, $n-1$ rovníc o $n-1$ neznámych. Na nájdenie zvyšných dvoch rovníc potrebujeme aj dve okrajové podmienky. Prvú rovnicu dostaneme z okrajovej podmienky $P(0,\tau)=1$, pre $i=0$ je hodnota P_0 rovná 1, teda vyhovuje rovnici (4.12). Druhú rovnicu dostaneme z okrajovej podmienky $P(\infty,\tau)=0$.

Použitím (4.11) a (4.12) dostaneme $n-1$ rovníc o $n-1$ neznámych, pre jednu časovú vrstvu j. Tento systém zapíšeme do maticového tvaru

$$AP^j = P^{j-1}b\,, \tag{4.19}$$

$j=1,2,...,m$, kde

- $b=(1-a_1,1,...,1)$ je počiatočný vektor
- $P^j=(P_1^j,...,P_{n-1}^j)$ je vektor neznámych hodnôt na časovej vrstve j
- $P^{j-1}=(P_1^{j-1},...,P_{n-1}^{j-1})$ je vektor známych hodnôt derivátu na časovej vrstve $j-1$
- $A=\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0\\ a_2 & b_2 & c_2 & & \vdots\\ 0 & a_3 & b_3 & \ddots & 0\\ \vdots & & \ddots & \ddots & c_{n-2}\\ 0 & \cdots & 0 & a_{n-1} & b_{n-1}\end{pmatrix}$

je trojdiagonálna matica rozmeru $(n-1)\times(n-1)$.

Riešením tohoto systému získame aproximácie hodnôt P na jednotlivých vstvách j a pre jednotlivé hodnoty úroku r.

4.4 Testovací program

V predchádzajúcich častiach sme si zadefinovali všetky pojmy potrebné v numerike. V tejto časti popíšeme základné črty nami vyhotoveného programu, jeho funkčnosť a postup pri jeho vytváraní. Program je vyhotovený v programovacom jazyku C, ktorý sme zvolili najmä vďaka jeho rýchlosti, pretože tú potrebujeme kvôli náročným výpočtom použitým v našej numerickej schéme.

Matematické objekty (vektor, matica), s ktorými narábame v numerickej schéme sú reprezentované dynamicky alokovanými poliami. Na začiatku programu si definujeme niekoľko konštánt, ktoré parametrizujú riešený problém. Postupne budeme popisovať ich význam. Prvou z nich je n, ktorá určuje, že počítame so systémom lineárnych rovníc rozmeru $n-1 \times n-1$.

Postup programu je nasledovný. Vytvoríme vektor r o rozmere n, ktorý predstavuje úrok. Jednotlivým zložkám vektora r pridelíme hodnoty podľa delenia priestorového intervalu. Na tento účel slúži funkcia $computeR(i)$, ktorá pre zadanú zložku i vypočíta príslušnú hodnotu, v našom prípade nerovnomerného delenia, ktoré je kombináciou troch lineárnych funkcií. Takéto delenie sme zvolili za účelom hustejšieho delenia v okolí θ (zadaná konštanta).

Následne vytvoríme tri vektory a, b, c o rozmere $n-1$ a maticu A o rozmere $n-1\times n-1$. Matica A bude trojdiagonálna, pričom jej hornú diagonálu predstavuje vektor c, dolnú diagonálu vektor a a diagonálu vektor b. Uvedené tri vektory naplníme na základe našej numerickej schémy so znalosťou konštánt $\kappa, \sigma, \beta, \lambda, \theta, T, k$ a vektora úroku r pomocou príslušných funkcií $computeA(i)$, $computeB(i)$ a $computeC(i)$.

Vytvoríme si počiatočný vektor v_0, z ktorého bude štartovať výpočet. V našom prípade je tento vektor jednotkový. Ešte si vytvoríme vektor reprezentujúci pravú stranu rovnice $Ax = b$. Tento vektor v je jednotkový až na svoju prvú zložku, ktorá podľa okrajovej podmienky je $1 - a_1$.

Hlavnou funkciou programu je funkcia $solve$, ktorá pre zadanú maticu A, vektor b a počiatočný vektor x v cykle pre $j = 1 \dots m$ (m je zadaná konštanta udávajúca počet časových rezov) počíta sústavu lineárnych rovníc 4.18. Na výpočet takejto sústavy sme implementovali Superrelaxačnú metódu (SOR) riešenia sústavy lineárnych rovníc.

Postup pri výpočte SOR metódy je nasledovný:

- riešime sústavu lineárnych rovníc v tvare $Ax = b$, kde matica A sa dá rozložiť na dolnú maticu L, hornú U a diagonálnu maticu D

- príslušnú iteračnú maticu T_ω volíme v tvare $T_\omega = (D + \omega L)^{-1}[-\omega U + (1 - \omega)D]$, tak aby $||T_\omega|| < 1$ ()

- zložky vektora x_i^{j+1} sa počítajú zo vzorca

$$x_i^{j+1} = \frac{\omega}{a_{ii}}(b_i - \sum_{k=1}^{i-1} a_{ik}x_k^{j+1} - \sum_{k=i+1}^{n} a_{ik}x_k^j) + (1 - \omega)x_i^j$$

V našom programe využívame funkciu *sor*, ktorá pre zadanú maticu, počiatočný vektor, vektor pravej strany a parameter ω vypočíta riešenie (vektor), ktoré spĺňa rovnicu v nami zadanej presnosti (zadaná konštanta *precision*). Parameter ω určuje rýchlosť konvergencie. Po skončení výpočtu funkcie *sor* dostávame vektor P^j, ktorý je j-tym časovým rezom. Na výpočet $j + 1$. časového rezu opäť spustíme funckiu *sor*, tentoraz s vektorom pravej strany P^j v prvej zložke zmenšenom o a_1 a počiatočným vektorom P^{j-1}. Týmto spôsobom získame všetkých m časových rezov riešenia.

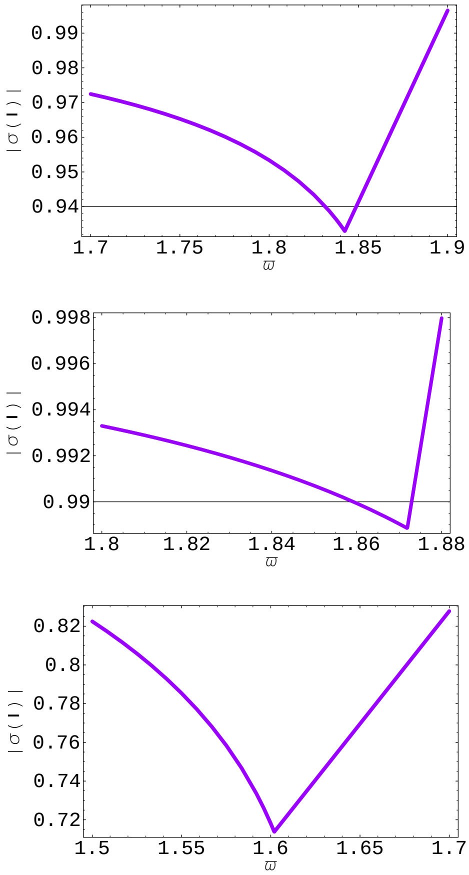
Na výpis vektora voláme funkciu *Vectorprint* a na výpis matice funkciu *Matrixprint*. Tieto funkcie dávajú riešenie na výstup vo formáte, ktorý je vhodný ako vstup pre program *Mathematica*, v ktorom sme dosiahnuté výsledky analyzovali a porovnávali s výsledkami explicitných modelov.

4.4.1 Voľba parametra ω

Pre každý z modelov uvedených v tretej kapitole je parameter ω iný. Tento parameter určuje rýchlosť konvergencie riešenia a volí sa z intervalu $\langle 0, 2\rangle$.

V programovacom jazyku Mathematica sme naprogramovali výpočet a vykreslenie parametra ω, ktorý je optimálny pre riešenie Vašíčkovho, CIR a Ho and Lee modela s rôznymi parametrami, ktoré vstupujú do výpočtu. Parameter ω podstatne urýchlil konvergenciu riešení.

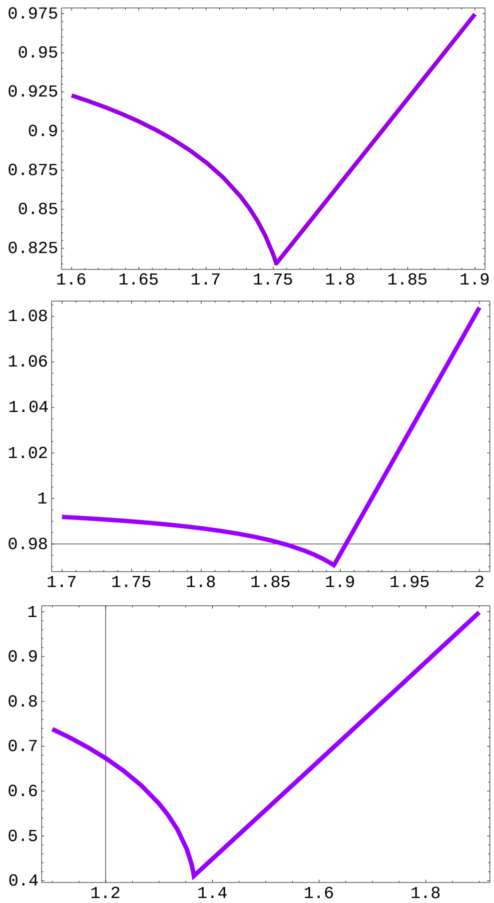
Vhodné parametre ω pre dané tri modely vidíme na nasledujúcom obrázku 3 ako minimum danej krivky.



Obr. 3: Parameter omega pre Vašíčkov, CIR a Ho and Lee model pre nasledujúce parametre: $\sigma = 0.4$, $\lambda = 0.8$, $\beta = 0.5, 0, 1$, $\theta = 0.06$.

Tabuľka 4.1.

	Vašíček	CIR	Ho and Lee	Vašíček	CIR	Ho and Lee
M	3650	365	3650	365	3650	365
σ	0.4	0.1	0.4	0.1	0.4	0.1
λ	0.8	0.5	0.8	0.5	0.8	0.5
β	0.5	0.5	0	0	1	1
ω	1.84	1.74	1.87	1.895	1.6	1.36



Obr. 4: Parameter omega pre Vašíčkov, CIR a Ho and Lee model pre nasledujúce parametre: $\sigma = 0.1$, $\lambda = 0.5$, $\beta = 0.5, 0, 1$, $\theta = 0.06$.

Na predchádzajúcich obrázkoch (obr. 3 a obr. 4) a v tab.4.1 je ukázané ako sa mení parameter ω za zmeny ostatných parametrov v daných troch modeloch. V podstate sa príliš nelíšia, napriek tomu je konvergencia riešení v konkrétnych modeloch najrýchlejšia s daným parametrom ω, ktorý je vypočítaný programom v Mathematice a priamo závisí na konkrétnej matici A a na voľbe ostatných konštánt. Pre každý model je určený práve jeden parameter ω, ten vidno v tabuľke a na obrázkoch.

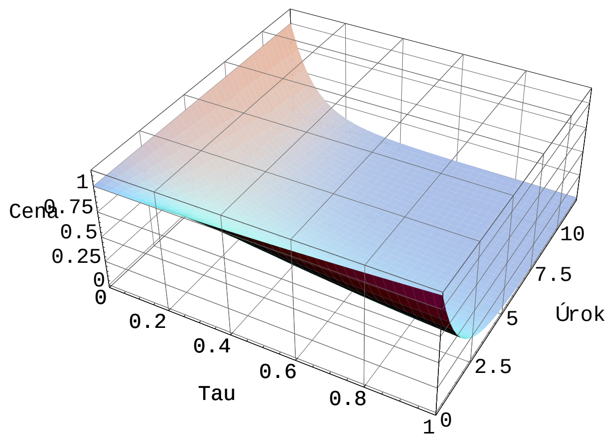
5 Testovanie výsledkov

V tejto kapitole sa budeme zaoberať najmä porovnávaním riešení, ktoré sme vypočítali pomocou našej numerickej schémy s riešeniami pomocou explicitných schém rovnovážnych a bezarbitrážnych modelov. Ukážeme si podstatné rozdiely a zistíme, ktoré riešenia sa viac približujú k reálnym výsledkom.

5.1 Porovnávanie riešení

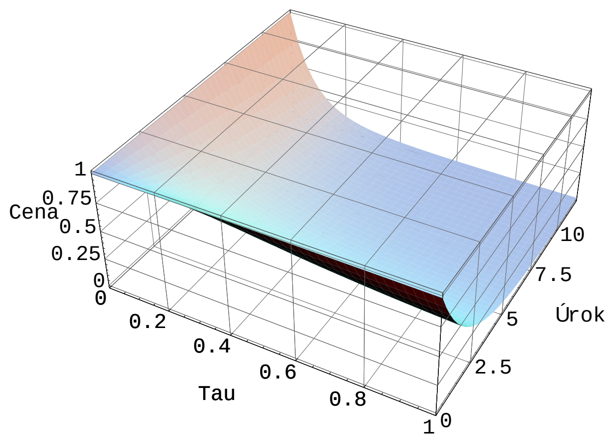
Ako sme už spomenuli budeme porovnávať rôzne riešenia. Na nasledujúcich obrázkoch uvidíme porovnané riešenia pre Vašíčkov, CIR, Ho and Lee model. Porovnávané riešenia sú vypočítané pre hodnoty derivátu za určitú časovú jednotku[6] a pre rôzne úroky[7].

Na týchto obrázkoch (obr. 5, obr. 6, obr. 7) vidíme grafické riešenia pre Vašíčkov, CIR, Ho and Lee model v trojdimenziálnom priestore. Vidieť, že majú podobný priebeh, aj keď ďalej sa ukáže, že tieto tri modely majú riešenia úplne rôzne.

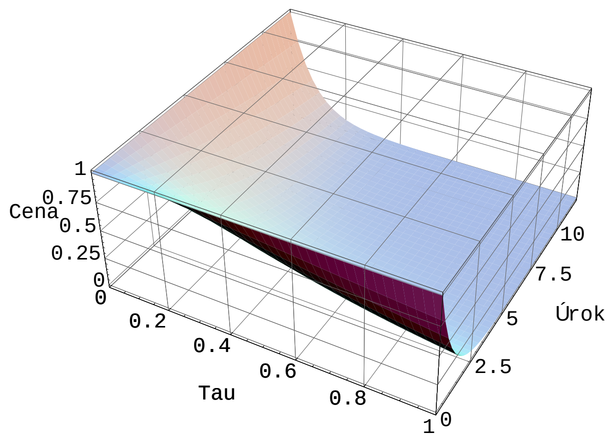


Obr. 5: Vašíčkov v trojdimenziálnom priestore.

[6]V našom prípade je to tzv. over night (jeden deň), dva týždne, mesiac, dva mesiace, tri mesiace, pol roka a rok.
[7]Nás najviac zaujímal úrok v okolí θ.



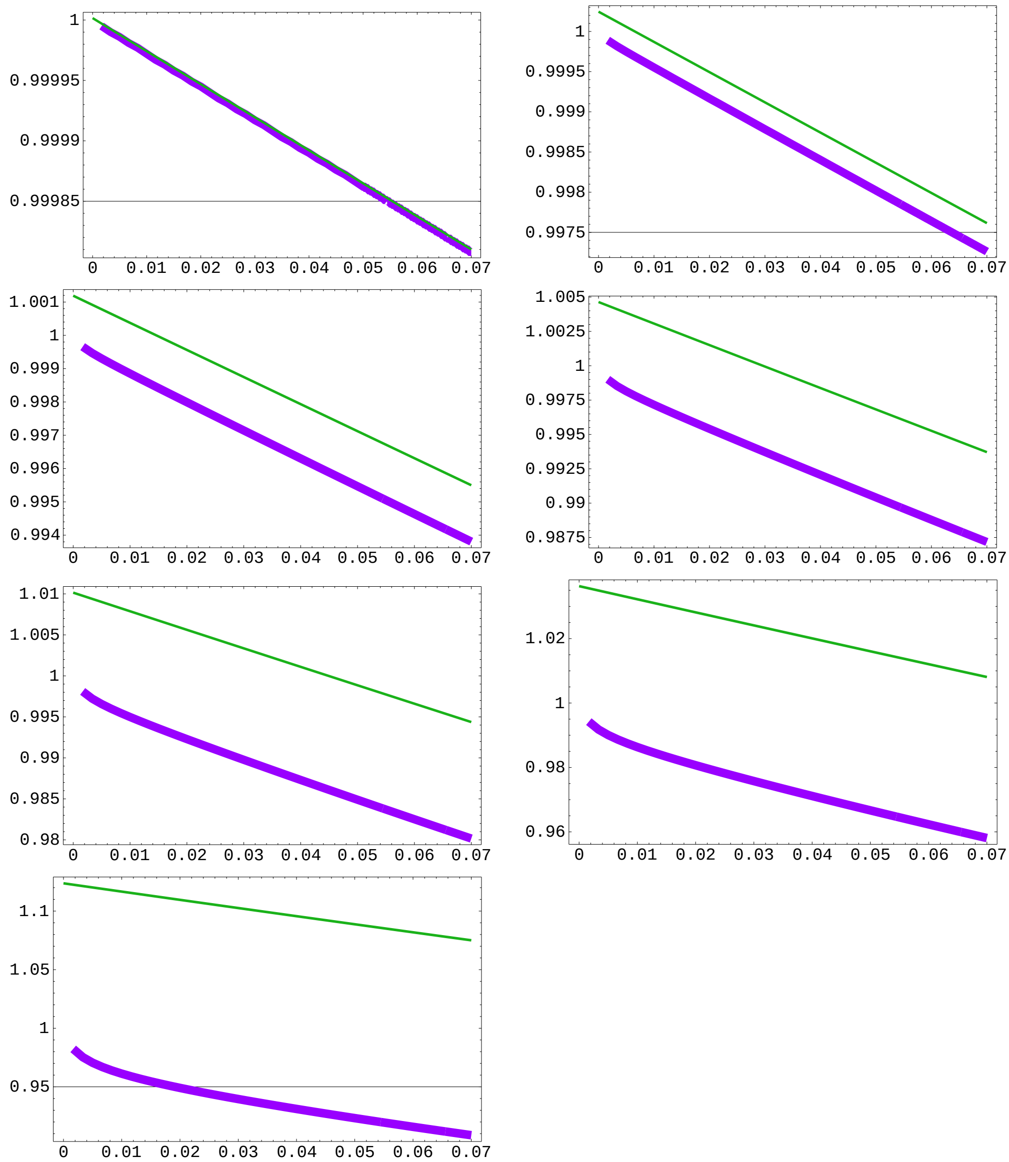
Obr. 6: CIR model v trojdimenziálnom priestore.



Obr. 7: HO and Lee model v trojdimenziálnom priestore.

Najskôr si rozoberieme Vašíčkov model. Na obrázku 8 ľahko vidieť, že riešenia sú s narastajúcim časom čoraz rozdielnejšie. Zatiaľ čo na začiatku obchodovania je hodnota derivátu skoro rovnaká pre numerickú schému, aj pre explicitné riešenie, na konci už vidieť podstatný rozdiel.

Riešenie Vašíčkovho modela počítaného z explicitného vzorca prekročí hneď po dvoch týždňoch stanovenú cenu dlhopisu a v reálnom svete cena dlhopisu nikdy neprekročí svoje maximum, ktoré sa stanoví hneď na začiatku obchodovania (v našom prípade je maximum rovné jednej). Riešenie, ktoré sme naprogramovali podľa našej numerickej schémy sa viac približuje reálnym možnostiam, pretože sme zvolili správnu okrajovú podmienku a stanovené maximum sa neprekročí a cena dlhopisu pozvoľna klesá. Teda riešenie, počítané



Obr. 8: Riešenia pre Vašíčkov model(riešenia hrubšou čiarou sú naše a tenšou sú podľa explicitného vzorca) pre parametre: čas = 1 deň, 2 týždne, 1, 2, 3 a 6 mesiacov, 1 rok, $\sigma = 0.4$, $\beta = 0.5$, $\theta = 0.6$, $\lambda = 0.8$, $\omega = 1.841$.

podľa explicitného vzorca nám nedáva také presné výsledky, pretože v ňom vychádzali z umelo vytvorených podmienok. Dôvodom zlyhania explicitného vzorca je práve nedodržanie správnej okrajovej podmienky. Tým, že sme si určili správnu podmienku v našej numerickej schéme, máme aj správnejšie výsledky, a zatiaľ môžeme tvrdiť, že je lepšia a spoľahlivejšia ako explicitné vzorce.

Teraz budeme porovnávať výsledky pre CIR model. Vidíme (obr. 9), že na začiatku je situácia podobná ako v predchádzajúcom prípade. S narastajúcim časom sa však riešenia zmenia. Aj v tomto prípade je to v prospech našej numerickej schémy. Riešenie počítané podľa našej numerickej schémy má prudší sklon ako riešenie vypočítané podľa explicitného vzorca. Smer krivky riešení z explicitného vzorca je veľmi nepravdepodobný. Najskôr má sklon aj tvar podľa našeho riešenia a potom zrazu začne klesať voľnejšie ako na začiatku obchodovania. To by znamenalo, že s narastajúcim úrokom veľmi voľne klesá cena dlhopisu, takmer nebadane. Aj keď sú výsledky riešenia z explicitného vzorca menej presné ako výsledky našej numerickej schémy, sú lepšie ako v prípade Vašíčka, pretože maximum dlhopisu, ktoré je stanovené na začiatku, prekročia len o kúsok.

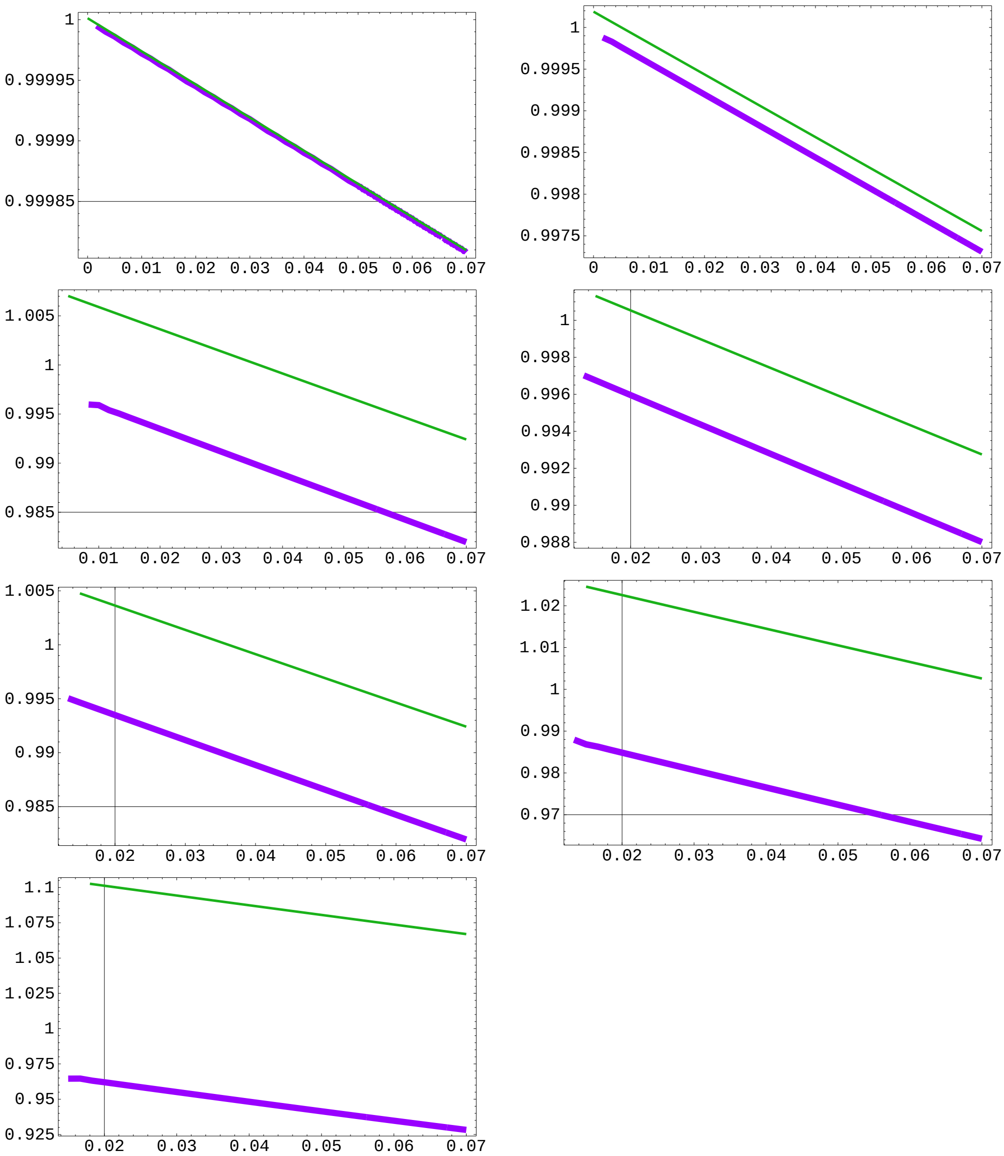
Ako posledný si rozoberieme Ho and Lee model. Z obrázka (10)vidieť, že riešenie počítané podľa explicitnej schémy je dobré, pretože Ho and Lee model zohľadňuje to, že tieto riešenia majú byť presne zhodné s dnešnou *term structure*. Okrem iného sa veľmi podobá na naše riešenie, až na odchýlky v čase tri mesiace, pol roka a jeden rok. V našej numerickej schéme nastáva mierna numerická oscilácia (aj to na intervale, ktorý nás veľmi nezaujíma), dôvodom je parameter β, ktorý ovplyvňuje najmä členy r_i^{β} a $r_i^{2\beta}$ v rovnici (4.17). Ak je hodnota parametra β blízka 1, alebo je $\beta > 1$, tak nastáva tento jav. Takýto problém sa dá odstrániť rôznymi spôsobmi. Napríklad v numerickej schéme oceňovania derivátov úrokovej miery rozdeliť interval $(0, \theta - d)$ na dva ďalšie intervaly, alebo zmenšiť časový krok. Obrázok 10 popisuje riešenia Ho and Lee modela po zmenšení časového kroku.

Na nasledujúcom obrázku 11 sú vykreslené spojito úročené úrokové miery v čase t s maturitou T vypočítané podľa rovnice (1.5).

Vo všeobecnosti môže mať krivka, charakterizujúca vzťah medzi výškou úrokovej miery a dobou splatnosti, rôzne tvary.

Za normálny tvar sa považuje rastúca krivka, vtedy sa očakáva nárast úrokových mier, čiže v budúcnosti budú úroky vyššie a cena dlhopisov poklesne. Ak je krivka klesajúca, v budúcnosti sa očakáva pokles úrokov a s tým nárast cien dlhopisov. Ak je krivka rastúca a potom mierne zahnutá doprava, tak v krátkom čase sa očakáva nárast úrokových mier, ale v budúcnosti zase ich pokles.

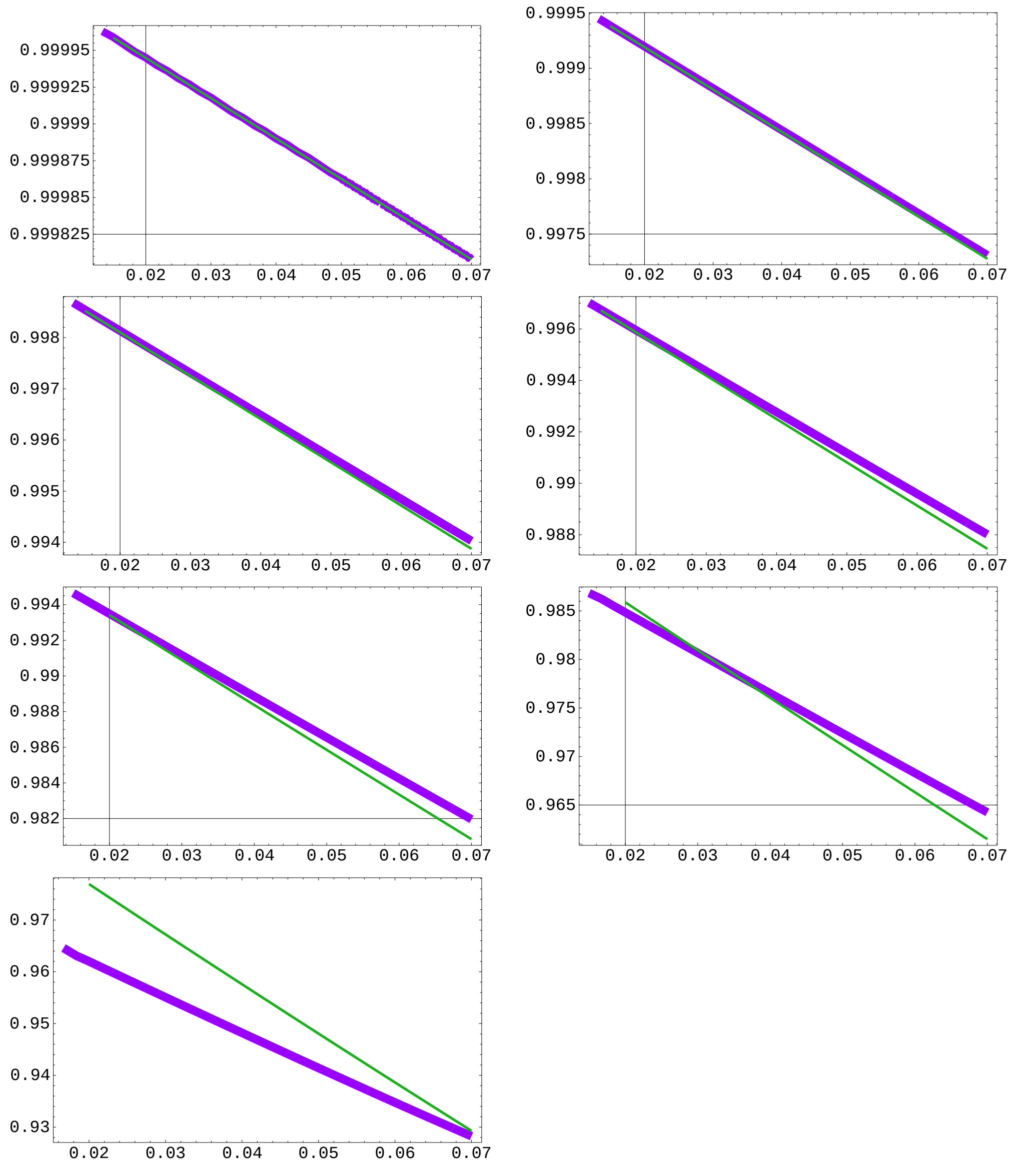
Nevýhoda Vašíčkovho modela počítaného z explicitného vzorca sa prejavuje tak, že dovoľuje aj záporné úroky a tie nemajú s realitou nič spoločné. CIR model podľa explicitných vzorcov síce nepovoľuje záporné úroky, ale napriek tomu má krivka klesajúci charakter.



Obr. 9: Riešenia pre Cox - Ingersoll - Ross model(riešenia hrubšou čiarou sú naše a tenšou sú podľa explicitného vzorca) pre parametre: čas = 1 deň, 2 týždne, 1, 2, 3 a 6 mesiacov, 1 rok, $\sigma = 0.4$, $\beta = 0$, $\theta = 0.6$, $\lambda = 0.8$, $\omega = 1.867$.

Najvernejšie opisuje reálnu situáciu Ho and Lee model, ktorý presne opisuje vývoj vzťahu úrokovej miery a doby splatnosti. Naše riešenia sa vo všetkých troch modeloch správajú podľa očakávaní, t.j. krivky majú rastúci priebeh, bez nejakých výkyvov.

Klesajúci priebeh výnosových kriviek Vašíčkovho a CIR modela značí o tom, že tieto modely nie sú korektné. Keďže už v okolí času $t = 0$ výnosová krivka klesá a to je hneď na začiatku obchodovania, v reálnom svete by sme asi nenašli investora, ktorý by bol

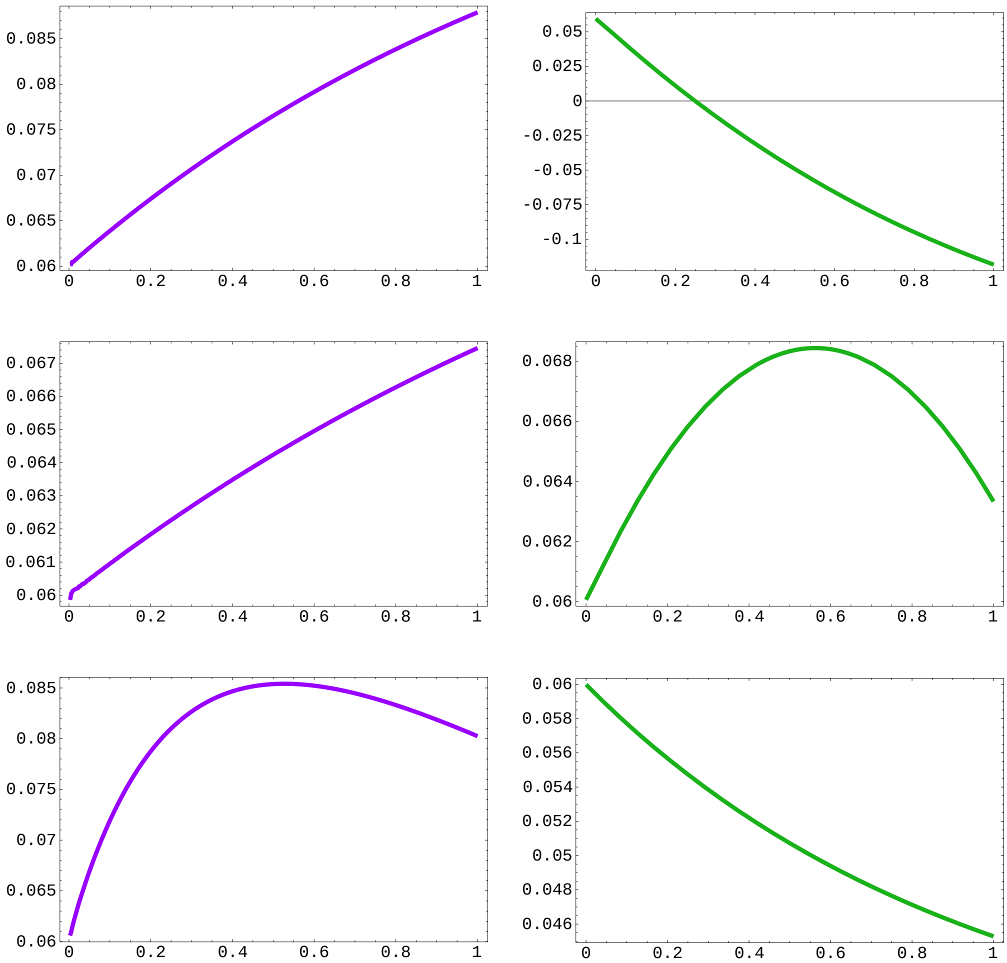


Obr. 10: Riešenia pre Ho and Lee model (hrubšou čiarou sú naše a tenšou sú explicitné) pre parametre: čas = 1 deň, 2 týždne, 1, 2, 3 a 6 mesiacov, 1 rok, $\sigma = 0.4$, $\beta = 1$, $\theta = 0.6$, $\lambda = 0.8$, $\omega = 1.6$.

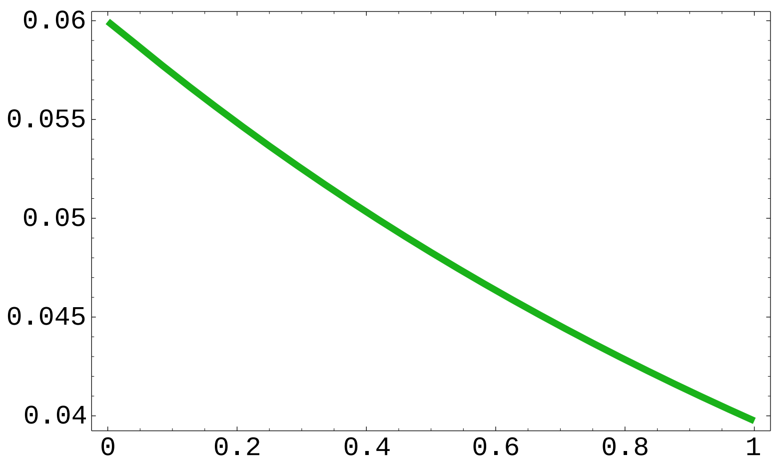
ochotný vlastniť takýto druh derivátu.

Vo Vašíčkovom modeli (obr. 12) sa záporná úroková miera dá odstrániť vhodnou voľbou parametrov σ a λ, klesajúci charakter krivky sa však touto voľbou neodstráni.

V našich riešeniach, ktoré zahŕňajú nové okrajové podmienky nemusíme nič odstraňovať, pretože výsledky sú korektné a vystihujúce situáciu na finančnom trhu. Týmto sa overila správnosť voľby nových, nami určených okrajových podmienok.



Obr. 11: Úrokové miery pre Vašíčkov, Ho and Lee, CIR model(vľavo je podľa našej numerickej schémy a vpravo je z explicitného vzorca).



Obr. 12: Výnosová krivka Vašíčkovho modela pre parametre: $\sigma = 0.1$, $\lambda = 0.5$

Záver

V predloženej diplomovej práci sme sa zaoberali numerickými metódami na oceňovanie derivátov úrokovej miery. Navrhli sme dve nové metódy na oceňovanie derivátov, v jednej sme použili rovnomerné delenie pre priestorový a aj časový interval a v druhej sme navrhli nerovnomerné delenie pre priestorový a rovnomerné delenie pre časový krok. Sú to nové metódy, ktoré vychádzajú z metódy sietí a aproximácie derivácií pomocou konečných diferencií.

V prvej časti práce sme popísali tradičné modely úrokovej miery v spojitom čase, ktoré reprezentujú jednofaktorové modely Vašíčkov a Cox, Ingersoll a Ross (CIR) model. Sú charakteristické jednoduchým použitím a ich riešenia sa dajú vyjadriť v analytickom tvare. Pridaním časovo závislých parametrov, ktoré presne modelujú súčasnú výnosovú krivku dostaneme model, ktorý je reprezentovaný bezarbitrážnym Ho and Lee modelom. Taktiež poskytuje riešenie v analytickej forme.

Jednofaktorové modely sa dajú vyjadriť v jednoduchej forme a ich použitie je jednoduché, ale napriek tomu neopisujú reálne očakávania na finančných trhoch. Preto sme navrhli program, ktorý by reálnu situáciu lepšie charakterizoval. Tento program by mohol mať uplatnenie aj v praxi, po prípadnej modifikácii, lebo náš program je navrhnutý pre bezrizikové finančné deriváty. Aj napriek tomu naše riešenia dávali veľmi dobré výsledky, presne podľa očakávaní. Bez väčších nepresností opisovali reálnu situáciu. Taktiež výnosové krivky našich numerických riešení boli v porovnaní s explicitnými riešeniami reálnejšie. Poukázali sme na to, že explicitné riešenia, vychádzajúce z nesprávnych okrajových podmienok, nedávajú presné výsledky. Riešenia, ktoré zahŕňajú nové okrajové podmienky, boli správne a situáciu na trhu charakterizovali veľmi výstižne.

Literatúra

[1] J. Babušíková, M. Slodička, J. Weisz: *Numerické metódy .*
Skriptá UK.

[2] D.Ševčovič: *Parciálne diferenciálne rovnice.*
Skriptá UK.

[3] E. Vitásek: *Numerické metódy.*
SNTL-Nakladatelství technické literatury, 1987.

[4] Y. K. Kwok: *Mathematical Models of Financial Derivatives.*
Springer Verlag, 1998.

[5] J. Hull: *Options, Futures and Other Derivative Securities.*
Prentice Hall, 1989.

[6] James I. Buchanan, Peter R. Turner: *Numerical methods and analysis .*
McGraw-Hill, 1992.

[7] Zuzana Vavrovičová: *Equilibrium and no - arbitrage models of the term structures.*
Diplomová práca, 1999.

Príloha

Program na numerický výpočet hodnoty derivátu v programe C++ podľa numerickej schémy (4.17):

```
  int N = 500;
  int M = 365;
  double K = 1;
  double T = 1.;
  double KAPPA = T/ (double) 2*M;
  double SIGMA = 0.4;
  double LAMBDA = 0.8;
  double BETA = 0.5;
  double THETA = 0.06;
  double PRECISION = 0.0000001;
  double OMEGA = 1.35;
  double R_MAX = 12;



double max (double x, double y)
{
    if ( x< y)
        return y;
    return x;
}
//---------------------------------------

Vector::Vector(int size)
{
  members = new long double*[1];
  for (int j = 0; j < size; j++)
    members[j] = new long double[size];
```

```
  this->size = size;

  for (int i = 1; i <= size; i++ )
    this->set(i,0);
}
//--------------------------------------

void Vector::set(int index, double number)
{
  this->members[0][index-1] = number;
}
//--------------------------------------

void Vector::set(Vector *v)
{
  for (int i = 1; i <= this->getSize(); i++)
    this->set(i, v->get(i));
}
//--------------------------------------

double Vector::get(int index)
{
  if (index <= 0 || index > this->getSize())
    return 0;

  return this->members[0][index-1];
}
//--------------------------------------

Vector* Vector::add(Vector *v)
{
  Vector *result = new Vector(size);
```

```
  for(int i = 1; i <= size; i++)
  {
    double number = this->get(i) + v->get(i);
    result->set(i, number);
  }

  return result;
}
//---------------------------------------

int Vector::getSize()
{
  return this->size;
}
//---------------------------------------

void Vector::print()
{
    printf("{ ");
    for (int i = 1; i <= this->size; i++)
    {
      printf("%f", this->get(i));
      if (i < this->size )
        printf(", ");
    }
    printf(" }\n");
}
//---------------------------------------

Matrix::Matrix(int rows, int columns)
{
```

```
  if (rows != columns)
    exit(-1);

  members = new long double*[3];
  for (int j = 0; j < 3; j++)
    members[j] = new long double[columns];

  this->setRowsCount(rows);
  this->setColumnsCount(columns);

  for (int i = 0; i < 3 ; i++)
    for (int j = 1; j <= columns; j++)
      this->set(j-i+1,j,0);
}
//----------------------------------------

void Matrix::print()
{
  printf("{ ");
  for(int i = 1; i <= rowsCount; i++)
  {
    printf("{ ");
    for(int j = 1; j <= columnsCount; j++)
    {
      printf("%f", this->get(i,j));
      if (j < columnsCount) printf(", ");
    }
    printf(" },\n");
  }
  printf("}; \n");
}
//----------------------------------------
```

```
int Matrix::getRowsCount()
{
  return this->rowsCount;
}
//--------------------------------------

int Matrix::getColumnsCount()
{
  return this->columnsCount;
}
//--------------------------------------

void Matrix::setRowsCount(int rows)
{
  this->rowsCount = rows;
}
//--------------------------------------

void Matrix::setColumnsCount(int columns)
{
  this->columnsCount = columns;
}
//--------------------------------------

Vector* Matrix::getRow(int row)
{
  Vector *v = new Vector(this->columnsCount);
  for(int i = 1; i <= columnsCount; i++)
  {
    v->set(i, this->get(row, i));
  }
```

```
  return v;
}
//--------------------------------------

Vector* Matrix::getColumn(int column)
{
  Vector *v = new Vector(this->rowsCount);
  for(int i = 1; i <= rowsCount; i++)
  {
    v->set(i, this->get(i, column));
  }
  return v;
}
//--------------------------------------

double Matrix::get(int row, int column)
{
  if (row <= 0 || row > this->getRowsCount() ||
      column <= 0 || column > this->getColumnsCount())
    return 0;

  if (row == column - 1)
    return this->members[0][column - 1];

  if (row == column)
    return this->members[1][column - 1];

  if (row == column + 1)
    return this->members[2][column - 1];

  return 0;
}
```

```
//---------------------------------------

void Matrix::set(int row, int column, double number)
{
  if (row <= 0 || row > this->getRowsCount() ||
      column <= 0 || column > this->getColumnsCount())
    return;

  if (row == column - 1)
    this->members[0][column - 1] = number;

  if (row == column)
    this->members[1][column - 1] = number;

  if (row == column + 1)
    this->members[2][column - 1] = number;
}
//---------------------------------------

bool Matrix::isSquare()
{
  return (this->rowsCount == this->columnsCount);
}
//---------------------------------------

bool Matrix::nonameCriteria()
{
  double sum;
  double coef;
  for (int i = 1; i <= this->columnsCount; i++)
  {
    sum = 0;
```

```
    for (int j = 1; j <= this->rowsCount; j++)
    {
      coef = max(this->get(j,i)/this->get(j,j),
                  -(this->get(j,i)/this->get(j,j)));

      if (i != j)
        sum = sum + coef;
    }

    if (sum >= 1 )
      return false;
  }
  return true;
}
//----------------------------------------

bool Matrix::columnCriteria()
{
  double sum;
  for (int i = 1; i <= this->columnsCount; i++)
  {
    sum = 0;

    for (int j = 1; j <= this->rowsCount; j++)
      if (i != j)
        sum = sum + max(this->get(j,i), -(this->get(j,i)));

    if (sum >= max(this->get(i,i), -(this->get(i,i))))
      return false;
  }
  return true;
```

```
}
//---------------------------------------

bool Matrix::rowCriteria()
{
  double sum;
  for (int i = 1; i <= this->rowsCount; i++)
  {
    sum = 0;

    for (int j = 1; j <= this->columnsCount; j++)
      if (i != j)
        sum = sum + max(this->get(i,j), -(this->get(i,j)));

    if (sum >= max(this->get(i,i), -(this->get(i,i))))
      return false;
  }
  return true;
}
//---------------------------------------


int generalSor(Matrix *a, Vector *b, Vector *x_0,
                double omega, double precision)
{
  if (!a->isSquare())
  {
    printf("Zadana matica nie je stvorcova!");
    return 0;
  }

  if (a->getRowsCount() != b->getSize() ||
```

```
    b->getSize() != x_0->getSize())
{
  printf("Zadana matica a vektory nie su rovnakej velkosti!");
  return 0;
}

int maxindex;
int n = x_0->getSize();

Vector *x = new Vector(n);
Vector *x_old = x_0;
double number = 0;
double sum1 = 0;
double sum2 = 0;
double currentPrecision = precision + 1;
int iterations = 0;

while (currentPrecision > precision && iterations<500)
{
  for (int j = 1; j <= n; j++)
  {
    sum1 = 0;
    for (int k = 1; k < j; k++)
      sum1 = sum1 + a->get(j,k)*x->get(k);

    sum2 = 0;
    for (int k = j + 1; k <= n; k++)
      sum2 = sum2 + a->get(j,k)*x_old->get(k);

    number = (omega/(a->get(j,j)))*(b->get(j) - sum1 - sum2)
              + (1-omega)*x_old->get(j);
```

```
      x->set(j, number);
    }




// vypocet max |Ax -b |

    currentPrecision = 0.;

    maxindex=0;

    for(int j=1; j<=n; j++)
    {
      sum1=0;
      for(int k=1; k<=n; k++)
        sum1=sum1+(a->get(j,k))*(x->get(k));

      double pomoc=b->get(j);

      if (currentPrecision < fabs(sum1- pomoc) )
      {
        maxindex=j;
        currentPrecision = fabs(sum1- pomoc);
      }
    }

    *x_old = *x;

    iterations++;
  }
```

```
  return iterations;
}
//---------------------------------------

int sor(Matrix *a, Vector *b, Vector *x_0,
        double omega, double precision)
{
  if (!a->isSquare())
  {
    printf("Zadana matica nie je stvorcova!");
    return 0;
  }

  if (a->getRowsCount() != b->getSize() ||
      b->getSize() != x_0->getSize())
  {
    printf("Zadana matica a vektory nie su rovnakej velkosti!");
    return 0;
  }

  int maxindex;
  int n = x_0->getSize();

  Vector *x = new Vector(n);
  Vector *x_old = x_0;
  double number = 0;
  double sum1 = 0;
  double sum2 = 0;
  double currentPrecision = precision + 1;
  int iterations = 0;

  while (currentPrecision > precision && iterations<500)
```

```
{
  for (int j = 1; j <= n; j++)
  {
    sum1=0.; sum2=0.;

    if(j>1)
      sum1 = a->get(j,j-1)*x->get(j-1);

    if(j<n)
      sum2 = a->get(j,j+1)*x_old->get(j+1);

    number = (omega/(a->get(j,j)))*(b->get(j) - sum1 - sum2) +
               (1-omega)*x_old->get(j);
    x->set(j, number);
  }

  // vypocet max |Ax -b |
  currentPrecision = 0.;
  maxindex=0;

  for(int j=1; j<=n; j++)
  {
    sum1=0;
    for(int k=1; k<=n; k++)
      sum1=sum1+(a->get(j,k))*(x->get(k));

    double pomoc=b->get(j);

    if (currentPrecision < fabs(sum1- pomoc) )
    {
      maxindex=j;
      currentPrecision = fabs(sum1- pomoc);
```

```
      }
    }

    *x_old = *x;
    iterations++;
  }

  return iterations;
}
//----------------------------------------

void solve(Matrix *A, Vector *b, Vector *x_initial, Vector *a)
{
  Vector *x_rightside = new Vector(b->getSize());
  Vector *x_init = new Vector(x_initial->getSize());
  x_rightside->set(b);

  x_init->set(x_initial);

  for(int i=1; i <= M; i++)
  {
    sor(A, x_rightside, x_init, OMEGA, PRECISION);
    printf("P[%d] = ",i);
    x_init->print();
    x_rightside->set(x_init);
    x_rightside->set(1, x_rightside->get(1) - a->get(1));
  }

  free(x_rightside);
  free(x_init);
}
//----------------------------------------
```

```
double computeA(Vector *r, int i)
{
  double r_im1 = r->get(i-1);
  if (i <= 1)
    r_im1 = 0.0000001;

  return (KAPPA / ( (r->get(i+1) - r_im1) )) * (K*(THETA - r->get(i)) +
            LAMBDA*SIGMA*pow(r->get(i),BETA)) -
            (KAPPA / ( (r->get(i+1) - r_im1)*( r->get(i) - r_im1 )  )*
            SIGMA*SIGMA*pow(r->get(i),2*BETA));
}
//---------------------------------------

double computeB(Vector *r, int i) {

  double r_im1 = r->get(i-1);
  if (i <= 1)
    r_im1 = 0.0000001;

  return 1 + KAPPA*r->get(i) + (KAPPA /
          ( (r->get(i+1) - r->get(i))*( r->get(i) - r_im1) )*
          SIGMA*SIGMA*pow(r->get(i),2*BETA));

}
//---------------------------------------

double computeC(Vector *r, int i) {

  double r_im1 = r->get(i-1);
  if (i <= 1)
    r_im1 = 0.0000001;
```

```
  return (-KAPPA / ( (r->get(i+1) - r_im1) )) *
          ( K * (THETA - r->get(i)) + LAMBDA*SIGMA*pow(r->get(i),BETA)) -
          (KAPPA / ( (r->get(i+1) - r_im1)*( r->get(i+1) - r->get(i)) )*
          SIGMA*SIGMA*pow(r->get(i),2*BETA));

}
//--------------------------------------

double computeR(int i) {
//konstanty, ktore urcuju hustotu delenia okolo THETY
  double d = 0.01;
  double p = 0.2;

  if (i < N*p)
    return (double) i * (THETA - d)/(N*p);

  if (i > N*(1 - p))
    return ((R_MAX - THETA - d)/p )*( (double) i/N - 1. + p) + THETA + d;

  return ( ( 2.*d /(1 - 2*p)))*(((double) i/N) - p) + THETA - d;
}



Matrix* generateMatrix(int rows, int columns, Vector *a,
                        Vector *b, Vector *c)
{
  Matrix *result = new Matrix(rows, columns);
  for(int i = 1; i <= rows; i++)
    for(int j = 1; j <= columns; j++)
    {
      if (j == i)
```

```
        result->set(i, j, b->get(i));
      if (j == i+1)
        result->set(i, j, c->get(i));
      if (j == i-1)
        result->set(i, j, a->get(i));
    }
  return result;
}
//-----------------------------------------

//-----------------------------------------
//------------  MAIN  -------------------
//-----------------------------------------


int main() {

  Vector *r = new Vector(N);
  for(int i = 1; i <= N; i++)
    r->set(i, computeR(i));

  r->set(N+1, R_MAX*((double)(N+1)/N)*((double)(N+1)/N));

  printf("r = ");
  r->print();

  Vector *a = new Vector(N-1);
  Vector *b = new Vector(N-1);
  Vector *c = new Vector(N-1);
  for(int i = 1; i <= N-1; i++)
  {
    a->set(i, computeA(r, i));
```

```
  b->set(i, computeB(r, i));
  c->set(i, computeC(r, i));
}

printf("a = ");
a->print();
printf("b = ");
b->print();
printf("c = ");
c->print();

Matrix *mmm = generateMatrix(N-1,N-1,a,b,c);

Vector *v = new Vector(N-1);
for (int i=2; i <= N-1; i++)
  v->set(i,1);

v->set(1,1-a->get(1));
Vector *v_0 = new Vector(N-1);
for (int i = 1; i <= N-1; i++)
  v_0->set(i,1);

printf("A =\n");
mmm->print();
printf("b =\n");
v->print();
printf("x[0] =\n");
v_0->print();

bool matrixIsOK = false;

if ((mmm->rowCriteria()) || (mmm->columnCriteria())
```

```
      || (mmm->nonameCriteria()))
  {
    printf("Zadana matica splna aspon jedno kriterium konvergencie\n");
    matrixIsOK = true;
  }

if (matrixIsOK)
  {
   solve(mmm, v, v_0, a);

   printf( "NN = %d; \n", N);
   printf( "M = %d; \n", M);
   printf( "K = %f; \n", K);
   printf( "KAPPA = %f; \n", KAPPA);
   printf( "SIGMA = %f; \n", SIGMA);
   printf( "LAMBDA = %f; \n", LAMBDA);
   printf( "BETA = %f; \n", BETA);
   printf( "THETA = %f; \n", THETA);
   printf( "PRECISION = %f; \n", PRECISION);
   printf( "OMEGA = %f; \n", OMEGA);

  }
  exit (0);
}
//---------------------------------------
```

Výpis programu v Mathematice pre vhodný výber parametra ω:

```
<< data1.txt;

n=Length[A[[1]]];

DD=Table[ Table[ If[ i == j, A[[i]][[i]], 0], {j,1,n}], {i,1,n}];
```

```
L=Table[ Table[ If[ i > j, A[[i]][[j]], 0], {j,1,n}], {i,1,n}];

U=Table[ Table[ If[ i < j, A[[i]][[j]], 0], {j,1,n}], {i,1,n}];


T[om_]:=Inverse[DD+om L].((1-om) DD - om U);

vl[om_]:=Max[Abs[Eigenvalues[T[om]]]];


graf=Plot[vl[om], {om,1.7,1.9},
        Frame->True,
        FrameLabel->{"\[CurlyPi]", "|\[Sigma](T)|"},
        PlotStyle->{Thickness[.01], RGBColor[0.6,0,1]}];
Display["omega.eps", graf, "EPS"];
```

Program v Mathematice na výpočet hodnoty derivátu a jeho vykreslenie podľa explicitného vzorca pre Vašíčkov model:

```
sol=NDSolve[{-y'[tau]==K*y[tau]-1,y[0]==0}, y[tau],{tau,0,T}];
VasB[tau_]=y[tau]/.sol[[1]];

sol1=NDSolve[{-y'[tau]==
        K*THETA*VasB[tau]*y[tau]-
        LAMBDA*SIGMA*VasB[tau]-SIGMA^2/2*VasB[tau],
      y[0]==1},y[tau],{tau,0,T}];

VasA[tau_]=y[tau]/.sol1[[1]];

cas=14; t=Max[0., T - KAPPA*cas]; Print["Vasicek"];Print[VasA[T-t]
Exp[ - VasB[T-t] r ]]
```

```
data=Table[ {r[[i]], P[cas][[i]]}, {i,1,NN(1-PP)}]; Print["data"];
Print[data]

graf1=
    ListPlot[data,
        PlotRange->All,
        Frame->True,

        PlotStyle->{Thickness[.01], RGBColor[0.6,0,1]},
        PlotJoined->True];

graf2=Plot[VasA[T-t] Exp[ - VasB[T-t] r ],{r,0,0.07},
    Frame->True,
        PlotStyle->{Thickness[.01], RGBColor[0.1,0.7,0.1]},
    PlotRange->All];



subor="porovnaj1.eps";porovnaj=Show[graf1,graf2];
Display[subor,porovnaj,"EPS"];
```

Program v Mathematice na výpočet a zobrazenie výnosovej krivky pre Vašíčkov model:

```
Urok=0.06;
For[i=1, i<=NN-1, 1,
    {
If[r[[i]]>= Urok, {index=i; i=NN-1; } , ];
                i++;}];
data=Table[
        { KAPPA*cas,
            -Log[ P[cas][[index]] ]/ (KAPPA*cas)}, {cas,1, 365}];

subor="urok2.eps" graf3=
    ListPlot[data,
```

```
        PlotRange->All,
        Frame->True,
        PlotStyle->{Thickness[.01], RGBColor[0.6,0,1]},
        PlotJoined->True];
Display[subor,graf3,"EPS"];


subor="urok2v.eps" graf4=
    Plot[-Log[VasA[tau] Exp[ - VasB[tau] Urok] ]/tau ,{tau,0,T},
    Frame->True,
        PlotStyle->{Thickness[.01], RGBColor[0.1,0.7,0.1]},
    PlotRange->All];
Display[subor,graf4,"EPS"];
```

Program v Mathematice na výpočet hodnoty derivátu pre CIR model:

```
sol=NDSolve[{-y'[tau]==SIGMA^2/2*(y[tau]^2)+
            (K+LAMBDA)*y[tau]-1,y[0]==0},
    y[tau],{tau,0,T}];
CIRB[tau_]=y[tau]/.sol[[1]];
sol1=NDSolve[{-y'[tau]==K*THETA*CIRB[tau]*y[tau],y[0]==1},
              y[tau],{tau,0,T}];
CIRA[tau_]=y[tau]/.sol1[[1]];
```

Program v Mathematice na výpočet hodnoty derivátu pre Ho and Lee model:

```
sol=NDSolve[{y'[tau]==1,y[0]==0},y[tau],{tau,0,T}];
HolB[tau_]=y[tau]/.sol[[1]];
sol1=NDSolve[{-y'[tau]==y[tau]*THETA*HolB[tau]-
      1/2*SIGMA^2*HolB[tau]^2*y[tau],
      y[0] == 1},y[tau],{tau,0,T}];
HolA[tau_]=y[tau]/.sol1[[1]];
```