Budapesti Corvinus Egyetem
Eötvös Loránd Tudományegyetem

Biztosítási és pénzügyi matematika MSc szak

Kvantitatív pénzügyek specializáció

Halász Kristóf

Pénzügyi opciók árazása és modell kalibráció neuronhálókkal

Konzulens:

Dr. Fáth Gábor

2025

Köszönetnyilvánítás

Ezúton szeretném megköszönni a témavezetőmnek, Dr. Fáth Gábornak a diplomamunka írása során rám fordított időt és a sok hasznos meglátását és ötleteit.

Köszönöm családomnak a rengeteg biztatást és támogatást, amelyet az egyetemi éveim alatt kaptam.

Tartalomjegyzék

Bevezetés

A dolgozatban különféle pénzügyi termékek árazásának replikálását vizsgálom neurális hálók segítségével. Egyes modellek és termékek esetén a pontos árazás a hagyományos numerikus módszerek használatával általában lassan és nagy számításigénnyel végezhető el. Ezáltal a modellek kalibrálása a gyakorlatban idő- és erőforrásigényes algoritmusokhoz vezet.

A neuronhálók kiértékelése viszont a struktúrájukból adódóan hatékonyan elvégezhető, valamint a *backpropagation* algoritmus segítségével a termék input paraméterei szerinti parciális deriváltak is ugyanezzel a komplexitással számolhatóak. Ezzel a módszerrel akár lényeges gyorsítás érhető el a modellek kalibrálásánál [1], vagy a kockázatok számításánál bizonyos termékek esetében.

Az 1 fejezetben röviden bemutatom a dolgozatban vizsgált pénzügyi termékeket. Majd ezután a 2 fejezetben ismertetem a Black-Scholes-Merton, valamint a Heston modellek főbb jellemzőit és az ezekben történő árazást. A 2.4 alfejezet a modellekből kapott differenciálegyenletek numerikus megoldására alkalmazható módszereket tárgyalja részleteiben.

A 3 fejezetben az előrecsatolt neurális hálók matematikai alapjait mutatom be, melyben szerepel többek között a *backpropagation*, mint automatikus differenciáló algoritmus és a hálók tanításához használt optimalizáló algoritmusok.

A 4 fejezetben az árakra és az implikált volatilitásra tanított neurális hálók eredményeit ismertetem. A neuronhálók tanítása során alkalmazok egy parciális deriváltak által definiált regularizációt, amelynek segítségével a termékek kockázatai is pontosan számolhatóak a neurális hálókkal. A 4.2 alfejezetben a tanított neuronhálókat a Heston modell paramétereinek kalibrálására használom, szintetikusan generált piacok esetében. A kalibrálás során figyelembe veszem a barrier opciók árát is, amellyel pontosabban modellezhető a jövőbeni volatilitás mosoly és ferdeség. [2]

1. Opciók

Az opciók olyan pénzügyi derivatív termékek, melyek a tulajdonosnak jogot biztosítanak arra, hogy adott mennyiségű alapterméket egy előre meghatározott áron és időpontban vagy időpontig megvásároljon vagy eladjon. A szerződés másik oldalán álló félnek - az opció kiírójának - kötelezettsége van az ügylet teljesítésére.

Az elmúlt évtizedekben a pénzügyi piacok fejlődésével az opcióspiac mérete nagy mértékben megnövekedett, mind a sztenderdizált tőzsdei és az OTC piacokon egyaránt. [3] Ezen termékek növekvő népszerűségének az egyik oka a rugalmasságuk, ugyanis összetett opciós pozíciók segítségével tetszőleges kifizetési függvény előállítható határértékként. Ezáltal egyaránt alkalmasak kockázatok fedezésére, spekulációs pozíciók felvételére és portfólió diverzifikálásra.

A két leggyakoribb opció a vételi jog és az eladási jog (továbbiakban call és put opció):

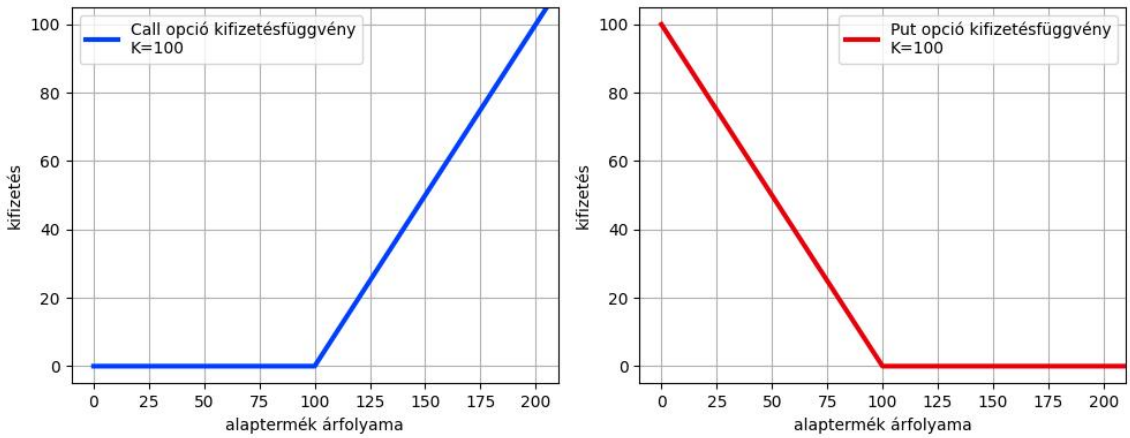
1.1. Definíció. *Európai típusú call és put opció: az opció tulajdonosának (long pozíció) joga van a szerződésben előre meghatározott lejáratkori időpontban (T) az adott mennyiségű alapterméket (S) a kötési árfolyamon (K) call opció esetén megvásárolni, put opció esetén eladni. Az opció kiírójának (short pozíció) kötelezettsége van az alapterméket call opció esetén eladni, put opció esetén megvásárolni a kötési árfolyamon lehívás esetén.*

A lejáratkori kifizetésfüggvény a long call opció esetében $max(0, S_T - K)$, long put opció esetében $max(0, K - S_T)$. Short pozícióknál a kifizetésfüggvény a long pozíció ellentettje.

Az opciók és egyéb derivatívák kiírói általában pénzügyi intézmények, amelyek a piacokon az árjegyző szerepét töltik be, ezáltal mindig elérhető az ár és lehetőség van a piacon kereskedni, fenntartva a likviditást. Az árjegyzők szempontjából tehát kulcsfontosságú a derivatívaárazás és ezzel együtt a megfelelő fedezeti stratégiának a kérdése: mennyi az a minimális kezdőtőke, amellyel kialakítható olyan fedezeti stratégia, amivel lejáratkor semmilyen esetben sem realizálható veszteség?

1.1. Barrier opciók

A barrier opciók olyan derivatívák, melyeknél az opciós szerződés megszűnik (vagy akkor lép életbe), ha az alaptermék árfolyama elér egy bizonyos szintet, a barriert (B). Két főbb



1. ábra. Call és put opció kifizetésfüggvénye

típusuk létezik,

1.2. Definíció. *Knock-out (KO) barrier opció: olyan opció, ahol a szerződés megszűnik, ha az alaptermék árfolyama* (S) *lejáratig eléri a barriert* (B)*. B és S viszonyától függően lehet "up-and-out" (UO), amikor $B > S$, vagy "down-and-out" (DO), ha $B < S$.*

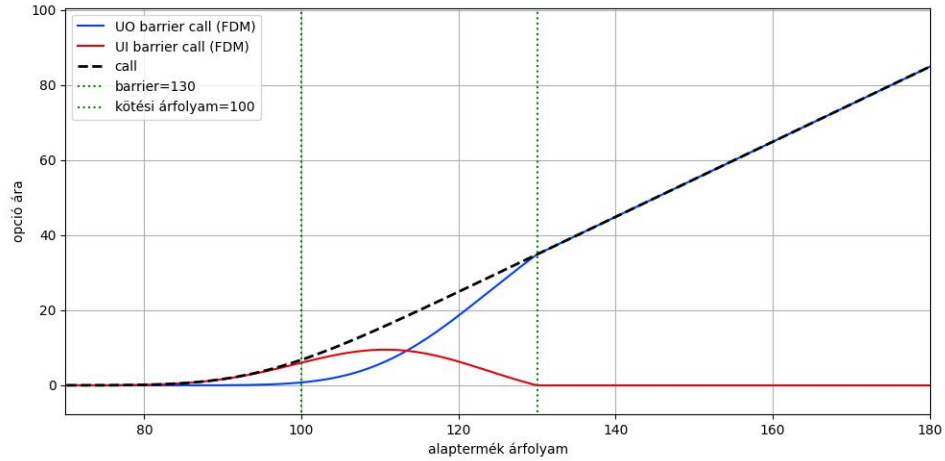
1.3. Definíció. *Knock-in (KI) barrier opció: olyan opció, ahol az opciós szerződés csak akkor lép életbe, ha az alaptermék árfolyama* (S) *lejáratig eléri a barriert* (B)*. B és S viszonyától függően lehet "up-and-in" (UO), amikor $B > S$, vagy "down-and-in" (DO), ha $B < S$.*

A barrier opciók trajektória függőek, továbbá a lejáratkori kifizetésfüggvényük nem folytonos a B pontban. Ezért a hagyományos call vagy put opció kifizetésfüggvénye szigorúan nagyobb, mint egy barrier call vagy put opció kifizetésfüggvénye. A barrier opciók tehát olcsóbbak, mint az azonos kötési árfolyamú call vagy put opció.

Call és put opciós árak mellett elegendő csak a KO vagy csak a KI típusú barrier opciókat árazni, ugyanis a call és a put opciók replikálhatóak az ugyanazzal a kötési árfolyammal rendelkező KO és KI barrier opciók összegével.

$$c_K = c^{UO}_{K,B} + c^{UI}_{K,B} = c^{DO}_{K,B} + c^{DI}_{K,B}$$
$$p_K = p^{UO}_{K,B} + p^{UI}_{K,B} = p^{DO}_{K,B} + p^{DI}_{K,B}$$

A 2 ábrán ugyanazzal a $K = 100$ és $B = 130$ paraméterű KO és KI barrier call opciók, valamint az összegükként előállítható call opció értéke szerepelnek az alaptermék árfolyamának függvényében.



2. ábra. Európai call és azonos kötési árfolyamú UO and UI barrier opciók értéke a Black-Scholes modellben, $T = 1, K = 100, B = 130, \sigma = 10\%, r = 5\%$

2. Árazási módszerek

2.1. Black-Scholes modell

Az opciók árára nem volt ismert analitikus megoldás egészen 1973-ig, amikor Fisher Black és Myron Scholes, valamint tőlük függetlenül Robert C. Merton előálltak árazási modelljükkel.[4][5] Ez az újféle megközelítés a Black-Scholes-Merton (BSM) modell néven híresült el. A legegyszerűbb változatban a modell feltételezései az alábbiak:

- Az alaptermék nem fizet osztalékot és árfolyama geometria Brown-mozgást követ:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t),$$

  ahol μ a drift és σ a volatilitás.

- Az eszközökkel folytonosan lehet kereskedni, lehetséges a shortolás.

- Nincsenek tranzakciós költségek és adók.

- Az r kockázatmentes kamatláb, valamint σ a volatilitás konstans.
- A piacon nincsen arbitrázs lehetőség.

A modell árazó formulájának levezetése részleteiben megtalálható John C. Hull *Options, futures and other derivatives* című könyvében.[6] Az Itô lemma segítségével fel tudjuk írni a derivatív termék f dinamikáját, majd az alaptermékből és a derivatív termékből olyan portfóliót képzünk melyben kiküszöböljük a Wiener-folyamatból eredő véletlent. Az arbitrázsmentesség feltételezés miatt, ekkor a portfólió infinitézimális hozama megegyezik a kockázatmentes hozammal. Differenciál alakban így a következő egyenlethez jutunk:

$$rf(t,S(t))dt = \frac{\partial f}{\partial t}(t,S(t))dt + rS(t)\frac{\partial f}{\partial x}(t,S(t))dt + \frac{1}{2}\sigma^2 S(t)^2 \frac{\partial^2 f}{\partial x^2}(t,S(t))dt. \qquad (1)$$

Ezt a egyenletet szokás Black-Scholes-Merton differenciálegyenletnek nevezni. Az egyenlethez tartozó peremfeltétel pedig a derivatíva lejáratkori kifizetésfüggvényével $f(T,S_T)$ adható meg.

Európai call és put opció esetén a megoldásra zárt formula adható, amely a nevezetes Black-Scholes (BS) formula:

- call opciónál a peremfeltétel $c(T,S_T) = max(0, S_T - K)$ és a zárt formula:

$$c(t,S_t) = S\Phi(d_1) - e^{-r(T-t)}K\Phi(d_2)$$

$$d_{1,2} = \frac{log(\frac{S_t}{K}) \pm (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$$

- put opciónál a peremfeltétel $p(T,S_t) = max(0, K - S_T)$ és a zárt formula a put-call paritást felhasználva:

$$p(t,S_t) = e^{-r(T-t)}K\Phi(-d_2) - S\Phi(-d_1)$$

A modell paramétereinek parciális deriváltjaira *görög betűk*ként *(option greeks)* szokás hivatkozni. Ezek közül is a legfontosabb a Δ, amely az alaptermék árfolyamához tartozó változó szerinti első parciális derivált. A replikáló stratégiában a Δ mutatja meg, hogy az

adott pillanatban hány darab alapterméket kell tartani a portfólióban. A piaccal együtt a Δ is folytonosan változik, így a replikáló stratégiát *dinamikus delta hedge*-nek is nevezik.

A volatilitás szerinti első parciális derivált a ν (görög nű betűvel jelölve, de vegának nevezve), a kockázatmentes hozam szerinti a ρ, a lejáratig hátralévő idő szerinti pedig a θ. Ezeket szokás a ν-nál és a ρ-nál bázispont vagy a θ-nál napi skálára hozni a könnyebb értelmezhetőség miatt.

A görög betűket általános termék esetén numerikusan szükséges közelíteni, viszont a BSM modellben call és put opció esetén a parciális deriváltakra van zárt formula, a standard normális eloszlás eloszlásfüggvénye (Φ) valamint sűrűségfüggvénye (φ) segítségével. A 1 táblázatban ezen értékeket szerepelnek call és put opciókra.

		Call	**Put**
Δ	$\partial f/\partial S$	$\Phi(d_1)$	$-\Phi(-d_1)$
ν	$\partial f/\partial \sigma$	$S\varphi(d_1)\sqrt{T-t}$	$S\varphi(d_1)\sqrt{T-t}$
θ	$\partial f/\partial t$	$-\frac{S\varphi(d_1)\sigma}{2\sqrt{T-t}} - rKe^{-r(T-t)}\Phi(d_2)$	$-\frac{S\varphi(d_1)\sigma}{2\sqrt{T-t}} + rKe^{-r(T-t)}\Phi(-d_2)$
ρ	$\partial f/\partial r$	$K(T-t)e^{-r(T-t)}\Phi(d_2)$	$-K(T-t)e^{-r(T-t)}\Phi(-d_2)$

1. táblázat. Analitikus formulák az európai call és put opciók görög betűkkel jelölt parciális deriváltjaira

2.2. Heston modell

A Heston modellt Márkus László *Pénzügyi folyamatok matematikája* előadásjegyzete [7], valamint Jim Gatheral *The volatility surface: a practitioner's guide* című könyve [2] alapján mutatom be.

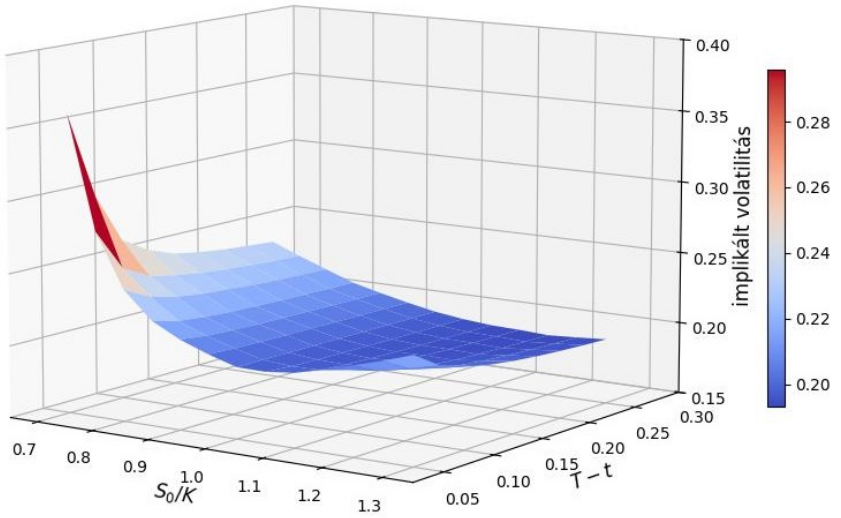
A Black-Scholes-Merton modellben a σ volatilitás amely az alaptermék árfolyamának paramétere konstans, és nem függhet a belőle származtatott derivatív termékek paramétereitől. Ceteris paribus, a Black-Scholes-Merton formulából számolt ár call és put opciókra a σ szigorúan monoton függvénye, ezt a függvényt jelöljük $f(\sigma)$-val. Ekkor $f(\sigma)$ invertálható és a piacon megfigyelt árakra definiálható az implikált volatilitás. Az implikált volatilitásra tekinthetünk úgy, mint a piacnak a volatilitásra vonatkozó jövőbeli várakozása.

2.1. Definíció. *Implikált volatilitás*

Legyen $c(T,K)$ a T időpontban lejáró K kötési árfolyamú európai call opció ára, ekkor azt a σ_{IV} volatilitást amelyre Black-Scholes-Merton formulából kapott ár megegyezik $c(T,K)$-val implikált volatilitásnak nevezzük.

Call és put opciók piacán megfigyelt árakból számolt implikált volatilitás a lejárat és a kötési árfolyam függvényében általában nem konstans, ezért a Black-Scholes-Merton modell nem írja le jól az alaptermék árfolyamának dinamikáját. Az implikált volatilitás felületen a K és a T függvényében is megfigyelhető nem nulla görbület, amelyet volatilitás mosolynak *(smile)*, az aszimmetriáját pedig ferdeségnek *(skew)* neveznek. Empirikusan a volatilitás mosoly és ferdeség időben dinamikusan változik, és a rövid lejáratokra a legmeredekebb.

További historikusan megfigyelt tulajdonságai a piacnak, amelyet a Black-Scholes-Merton formula nem képes tükrözni az ún. áttétel hatás és a volatilitás klasztereződés. Az előbbi azt jelenti, hogy az σ_{IV}és az alaptermék árfolyamának idősorai között erősen negatív a korreláció, azaz piaci esésnél az opciók implikált volatilitása megnő és fordítva. Az utóbbi pedig azt, hogy nagyobb változásokat a volatilitásban, hasonlóan nagy változások követnek. Az 3 ábrán egy Heston modell által generált implikált volatilitás felület látható, amelyen megfigyelhető a kötési árfolyamtól és a lejárattól függő mosoly és ferdeség.



3. ábra. Heston modell által generált implikált volatilitás felület rövid lejáratokra, $S_0 = 1.0, \kappa = 2.0, \theta = 0.04, \eta = 0.2$.

Ezen megfigyelések és tulajdonságok miatt több későbbi modellbe is beépítették azt a feltételezést, hogy a volatilitás legyen időfüggő és véletlen. Ezt a modellcsaládot sztochasztikus volatilitásmodelleknek nevezzük, és ebbe a családba tartozik a Heston modell is.

A Heston modellben két forrása is van a véletlennek, az alaptermék árfolyamát, valamint a sztochasztikus variancia folyamatot meghajtó Wiener-folyamat. Ezek között az áttétel hatás modellezése miatt szokás korrelációt feltételezni. A Heston modellben az alaptermék árfolyama az alábbi sztochasztikus differenciálegyenlet-rendszernek tesz eleget a kockázatsemleges mérték alatt

$$\begin{aligned} dS(t) &= rS(t)dt + S(t)\sqrt{V(t)}dW(t) \\ dV(t) &= \kappa(\theta - V(t))dt + \eta\sqrt{V(t)}d\hat{W}(t) \\ d\left[W, \hat{W}\right](t) &= \rho dt \\ S(0) &= S_0, \\ V(0) &= V_0. \end{aligned}$$

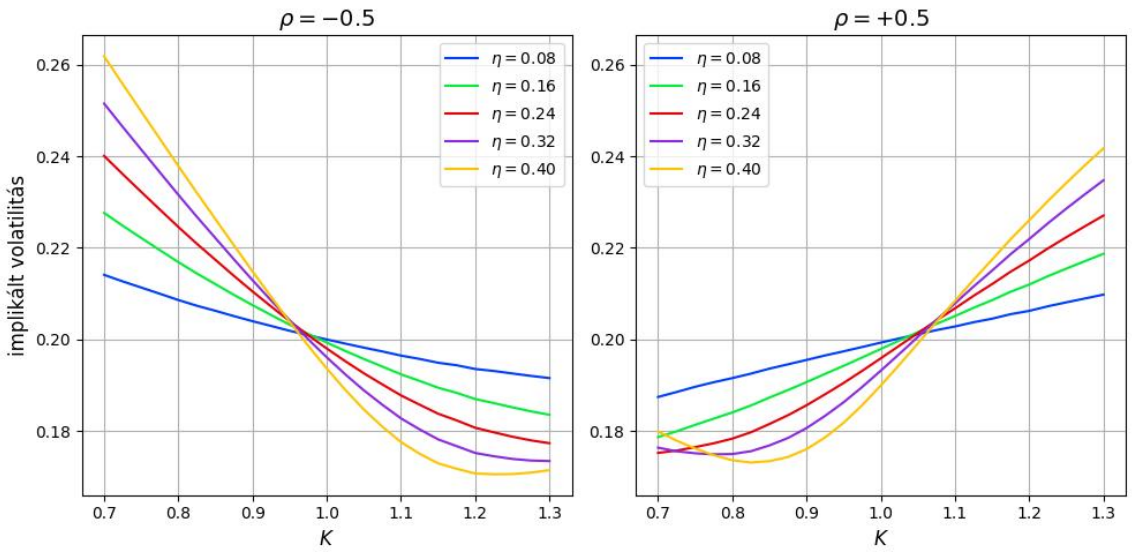
A sztochasztikus variancia folyamat egy CIR folyamat, melynek megoldásának létezését és pozitív értékkészletét Feller 2.1 tétele biztosítja.

2.1. Tétel. *Feller A $\kappa, \theta, \eta, v_0 > 0$ paraméterek mellett a*

$$\begin{aligned} dV(t) &= \kappa(\theta - V(t))dt + \eta\sqrt{V(t)}d\hat{W}(t) \\ V(0) &= V_0. \end{aligned}$$

egyenletnek létezik egyértelmű erős megoldása, melyre $V(t) \geq 0$. A $2\kappa\theta > \eta^2$ (Feller-feltétel) feltétel mellett a megoldás szigorúan pozitív.

A sztochasztikus variancia folyamatban a κ és a θ az átlaghoz visszahúzás sebességét és szintjét határozza meg, az η paraméter - amelyet volatilitás volatilitásának is szoktak nevezni - a mosoly görbületének mértékére van hatással. A Heston modellben a két Wiener-folyamat közötti ρ korrelációval befolyásolható a ferdeség, vagyis a volatilitás mosoly aszimmetriája. Az 4 ábrán az η és a ρ paraméterek hatása szerepel a modellben.



4. ábra. Implikált volatilitás a kötési árfolyam függvényében $T = 0.5$ lejáratra, az η és ρ paraméterek változtatása mellett. $S_0 = 1.0, \kappa = 2.0, \theta = 0.04, V_0 = 0.04$.

Egy derivatíva értékére a Black-Scholes-Merton egyenlethez analóg differenciálegyenlethez juthatunk a Feynman-Kac formula Itô-diffúziókra vonatkozó általánosításának segítségével.

2.2. Tétel. *Feynman-Kac formula Itô-diffúziókra*

Legyenek $v : \mathbb{R}^d \to \mathbb{R}$ *és* $k : \mathbb{R}^d \to \mathbb{R}^+$ *folytonos függvények, és tegyük fel, hogy az*

$$f : [0, T) \times \mathbb{R}^d \to \mathbb{R}$$

függvény $C^{1,2}$*-beli és kielégíti a*

$$-\frac{\partial}{\partial t} f(t, \underline{x}) + k(\underline{x}) f(t, \underline{x}) = \frac{1}{2} \boldsymbol{A}_x f(t, \underline{x})$$

egyenletet az $f(T, \underline{x}) = v(\underline{x})$ *peremfeltétellel. Ekkor teljesül, hogy*

$$f(t, \underline{x}) = \mathbb{E}_{\underline{x}} \left[v(\underline{w}(T - t)) \cdot exp \left\{ - \int_0^{T-t} k(\underline{w}(s)) ds \right\} \right],$$

ahol $\mathbb{E}_{\underline{x}}$ *azt jelenti, hogy egy eltolt Wiener-folyamat - amely* 0 *helyett* $\underline{x}$*-ből indul - szerinti várható értéket veszünk. Továbbá ez a megoldása a Cauchy-feladatnak egyértelmű.*

A tételben szereplő $\mathbf{A}_x$ legyen egy alaptermék árfolyam dinamikájának infinitézimális generátora az $\underline{x}$ változóban, valamint a magfüggvény legyen $k(\underline{x}) \equiv r$. Ekkor az alaptermék árfolyamának Markov tulajdonságát kihasználva az $\mathbb{E}_{\underline{x}}$ várható érték, pontosan az általános árazó formulával egyezik meg. Tehát a tételben szereplő differenciálegyenletet kielégítő $f(t, \underline{x})$ függvény a $v(\underline{x})$ kifizetési függvénnyel megadott derivatíva értéke a t időpontban, ha az alaptermék árfolyama pillanatnyilag t-ben $\underline{x}$.

2.3. Tétel. *Itô-diffúziók infinitézimális generátora*
Legyen

$$d\underline{Y}(t) = a(\underline{Y}(t))dt + \sigma(\underline{Y}(t))d\underline{W}(t)$$

d-dimenziós Itô-diffúzió, ahol $a : \mathbb{R}^d \to \mathbb{R}^d$ *és* $\sigma : \mathbb{R}^d \to \mathbb{R}^d$. *Ekkor* Y *az* $\boldsymbol{A}$ *infinitézimális generátora* $f \in C^{1,2}$ *függvényekre*

$$(\boldsymbol{A}f)(\underline{x}) = \sum_i \left(a_i(\underline{x}) \frac{\partial}{\partial x_i} f(\underline{x}) \right) + \frac{1}{2} \sum_{i,j} \left((\sigma\sigma^T)_{i,j}(\underline{x}) \frac{\partial^2}{\partial x_i \partial x_j} f(\underline{x}) \right).$$

A Heston modell alaptermék árfolyama két dimenziós Itô-diffúzió, és az infinitézimális generátorra vonatkozó tétel alapján

$$\mathbf{A}_{S,V} = rS\frac{\partial}{\partial s} + \kappa(\theta - V)\frac{\partial}{\partial v} + \frac{1}{2}VS^2\frac{\partial^2}{\partial s^2} + \frac{1}{2}\eta^2 V\frac{\partial^2}{\partial v^2} + \rho\eta VS\frac{\partial^2}{\partial v \partial s}.$$

Tehát a Heston modellben egy derivatív termék - melynek kifizetésfüggvénye a lejáratkori T-ben $v(S, V)$ - ára a Feynman-Kac formula alapján kielégíti a

$$rf = \frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}VS^2\frac{\partial^2 f}{\partial S^2} + \kappa(\theta - V)\frac{\partial f}{\partial V} + \frac{1}{2}\eta^2 V\frac{\partial^2 f}{\partial V^2} + \rho\eta VS\frac{\partial^2 f}{\partial V \partial S} \tag{2}$$

$$f(T, S, V) = v(S, V)$$

differenciálegyenletet, ahol az f-et mindenhol a (t, S, V) argumentumban kell kiértékelni.

A Heston modellnél a görög betűk a ν kivételével ugyanazokat a parciális deriváltakat jelölik, mint a BSM modellben. A ν értéke itt a volatilitás helyett a sztochasztikus variancia kezdeti értéke, vagyis a V_0 szerinti paricális deriváltat jelöli.

Egyes derivatív termékeknél, mint például az európai call opció a differenciálegyenlet

megoldható szemianalitikusan Fourier transzformáció segítségével.[2] Olyan termékeknél, ahol nem ismert gyors szeminanalitikus megoldás, az árat meghatározására a legelterjedtebb módszerek a Monte-Carlo szimuláció 2.3 és a parciális differenciálegyenlet numerikus megoldása 2.4.4.

2.3. Monte-Carlo szimuláció

Ebben az alfejezetben a Milstein módszert mutatom be röviden a Heston modellen [7], melyet a 2.4.4 alfejezetben bemutatott ADI módszer ellenőrzésére használok.

A Monte-Carlo szimulációhoz a sztochasztikus differenciálegyenlet diszkretizálása szükséges. Egyik lehetséges mód a Milstein módszer, amely az Itô-lemma segítségével határozza meg a diszkrét felosztás pontjai között a növekményeket. Tekintsük az alábbi általános Itô-diffúziót

$$dS(t) = a(S(t))dt + b(S(t))dW(t)$$

amely sztochasztikus integrál alakban írva

$$S(t+dt) = S(t) + \int_t^{t+dt} a(S(s))ds + \int_t^{t+dt} b(S(s))dW(s).$$

Alkalmazva az Itô-lemmát $a(S(t))$-re és $b(S(t))$-re, az

$$a(S(s)) = a(S(t)) + \int_t^s a'(S(u))a(S(u)) + \frac{1}{2}a''(S(u))b^2(S(u))du + \int_t^s a'(S(u))b(S(u))dW(u)$$

és

$$b(S(s)) = b(S(t)) + \int_t^s b'(S(u))a(S(u)) + \frac{1}{2}b''(S(u))b^2(S(u))du + \int_t^s b'(S(u))b(S(u))dW(u)$$

egyenletekhez jutunk. Visszahelyettesítve az eredeti differenciálegyenlet sztochasztikus integrál alakjába, és elhagyva a limeszben eltűnő tagokat az

$$S(t+dt) = S(t) + \int_t^{t+dt} a(S(t))ds + \int_t^{t+dt} b(S(t))dW(s) + \int_t^{t+dt}\int_t^s b'(S(u))b(S(u))dW(u)dW(s)$$

alakot kapjuk. Az utolsó tagot közelíthetjük

$$\int_t^{t+dt}\int_t^s b'(S(u))b(S(u))dW(u)dW(s) \approx \frac{1}{2}b'(S(t))b(S(t))\left[(\Delta W(t))^2 - dt\right]$$

módon, így a diszkretizálás az alábbi egyenlet alapján elvégezhető a kezdeti feltételből elindítva

$$S(t+dt) = S(t) + a(S(t))dt + b(t)\sqrt{dt}Z + \frac{1}{2}b'(S(t))b(S(t))dt(Z^2-1),$$

ahol Z standard normális valószínűségi változó.

A Heston modellben $V(t)$-re, majd $S(t)$-re is elvégezve a diszkretizálást, az alábbi egyenletekhez jutunk

$$V(t+dt) = V(t) + \kappa(\theta - V(t))dt + \eta\sqrt{V(t)dt}Z_V + \frac{1}{4}\eta^2 dt(Z_V^2 - 1)$$
$$S(t+dt) = S(t) + rS(t)dt + \sqrt{V(t)dt}S(t)Z_S + \frac{1}{4}S(t)^2 dt(Z_S^2 - 1)$$

ahol $Z_V = Z_1, Z_S = \rho Z_1 + \sqrt{1-\rho^2}Z_2$ és Z_1, Z_2 független standard normális valószínűségi változók. A diszkretizálás miatt a sztochasztikus varianciafolyamat fel tud venni akár negatív értéket is, ennek az esetnek a kezelésére lehetséges megoldás a teljes csonkolás módszere, amely a negatív értékeket 0-val helyettesíti. Másik megoldás a negatív érték helyettesítése az abszolút értékével.

Az ár meghatározásához az N darab szimulált trajektórián behelyettesítünk a derivatíva kifizetésfüggvényébe, és vesszük ezen értékek átlagát.

2.4. Véges differenciák módszere

Olyan zárt képlet, mint a BS formula az európai call és put opcióknál, nem minden derivatív esetén ismert. Barrier call és put opciók esetén viszont szintén léteznek analitikus formulák a BSM modellben. [8]

Egy alternatív árazási módszer, a parciális differenciálegyenlet numerikus megoldása viszont alkalmazható általános kifizetésfüggvénnyel rendelkező termékre is. A BSM modellben ez a 1 differenciálegyenlet megoldását jelenti egy lejáratig hátralévő idő - alaptermék árfolyam rácson. Ebben az alfejezetben bemutatott numerikus módszerek részleteiben megtalálhatóak a *Paul Wilmott on Quantitative Finance* című könyvben. [9]

A rács általában az idő tengelyen az árazási időtől ($t = 0$) a lejáratig ($T = m\Delta t$), az alaptermék árfolyamának tengelyén pedig 0-tól egy elegendően nagy értékig tart ($n\Delta S$). A $t = 0$ időpontban az árra van szükségünk, ezért a rács többi oldalán peremfeltételeket kell meghatározni. Lejáratkor a kifizetésfüggvény értékeit tudjuk használni, az alaptermék árfolyam tengelyének határainál pedig általában tudunk közelíteni valamilyen termékspecifikus függvény segítségével. Legyen az árazásra használt rács az egyes tengelyek mentén egyenlően felosztott Δt és ΔS lépésközökkel, a rácspontok halmaza legyen

$$\{0, \Delta t, 2\Delta t, \ldots, m\Delta t\} \times \{0, \Delta S, 2\Delta S, \ldots, n\Delta S\},$$

és az $(i\Delta t, j\Delta S)$ rácsponton az opció értékét jelöljük $f(i\Delta t, j\Delta S) = f_i^j$-vel.

2.4.1. Explicit véges differenciák módszere

Az explicit vagy hátrafelé-Euler módszernél azt feltételezzük, hogy a i-edik időpontban az alaptermék árfolyama szerinti parciális deriváltak megegyeznek a - már ismert - $i + 1$-edik időpontban közelített értékekkel, így a parciális deriváltakat numerikusan közelíthetjük:

$$\begin{aligned}
\frac{\partial f_i^j}{\partial t} &= \frac{f_{i+1}^j - f_i^j}{\Delta t} \\
\frac{\partial f_i^j}{\partial S} &= \frac{f_{i+1}^{j+1} - f_{i+1}^{j-1}}{2\Delta S} \\
\frac{\partial^2 f_i^j}{\partial S^2} &= \frac{f_{i+1}^{j+1} - 2f_{i+1}^j + f_{i+1}^{j-1}}{\Delta S^2}
\end{aligned}$$

Az f_j^i értékét a fenti közelítésekkel a 1 differenciálegyenlet szerint fel tudjuk írni az alábbi módon:

$$\begin{aligned} rf_i^j &= \frac{\partial f_i^j}{\partial t} + rj\Delta S\frac{\partial f_i^j}{\partial S} + \frac{1}{2}\sigma^2 j^2 \Delta S^2 \frac{\partial^2 f_i^j}{\partial S^2} = \\ &= \frac{f_{i+1}^j - f_i^j}{\Delta t} + rj\Delta S\frac{f_{i+1}^{j+1} - f_{i+1}^{j-1}}{2\Delta S} + \frac{1}{2}\sigma^2 j^2 \Delta S^2 \frac{f_{i+1}^{j+1} - 2f_{i+1}^j + f_{i+1}^{j-1}}{\Delta S^2}, \end{aligned} \tag{3}$$

amelyből f_i^j-t kifejezve az

$$f_i^j = a_j f_{i+1}^{j+1} + b_j f_{i+1}^j + c_j f_{i+1}^{j-1}$$

alakot kapjuk, ahol

$$\begin{aligned} a_j &= \frac{1}{1+r\Delta t}\frac{1}{2}\left(rj + \sigma^2 j^2\right) \\ b_j &= \frac{1}{1+r\Delta t}\left(1 - \Delta t\sigma^2 j^2\right) \\ c_j &= \frac{1}{1+r\Delta t}\frac{1}{2}\left(-rj + \sigma^2 j^2\right). \end{aligned}$$

Az f_0^j értékeket, vagyis $t = 0$ időbeli opciós árakat meg tudjuk határozni a peremfeltételből - amely a lejáratkori kifizetésfüggvény - iteratívan hátrafelé haladva.

Az explicit módszer numerikusan csak feltételesen stabil [9], ami azt jelenti, hogy Δt és ΔS rossz megválasztása esetén a módszer nem konvergens. A konvergencia feltétele az 1 differenciálegyenlet esetében, ha

$$\Delta t \leq \frac{1}{\sigma^2 n^2}.$$

2.4.2. Implicit véges differenciák módszere

Az implicit módszernél a parciális deriváltakat az i-edik időpontban közelítjük, ami az 1 egyenletnél az alábbi alakban írható fel:

$$rf_i^j = \frac{f_{i+1}^j - f_i^j}{\Delta t} + rj\Delta S\frac{f_i^{j+1} - f_i^{j-1}}{2\Delta S} + \frac{1}{2}\sigma^2 j^2 \Delta S^2 \frac{f_i^{j+1} - 2f_i^j + f_i^{j-1}}{\Delta S^2}. \tag{4}$$

Hátrafelé haladva az i-edik időpontbeli opciós árakat úgy határozzuk meg, hogy az ott

közelített parciális deriváltakatból visszakapjuk a már ismert $i+1$-edig időpontbeli árakat.

Az 1 differenciálegyenletnél a következő lineáris egyenletrendszert kell megoldani minden i időpontban:

$$\underbrace{\begin{bmatrix} a_{n-1} & b_{n-1} & c_{n-1} & 0 & \dots & & & & 0 \\ 0 & a_{n-2} & b_{n-2} & c_{n-2} & \dots & & & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_2 & b_2 & c_2 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & a_1 & b_1 & c_1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} f_i^n \\ f_i^{n-1} \\ \vdots \\ f_i^1 \\ f_i^0 \end{bmatrix}}_{\underline{f_i}} = \underbrace{\begin{bmatrix} f_{i+1}^n \\ f_{i+1}^{n-1} \\ \vdots \\ f_{i+1}^1 \\ f_{i+1}^0 \end{bmatrix}}_{\underline{f_{i+1}}}$$

ahol az

$$\begin{aligned} a_j &= -rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t, \\ b_j &= 1 + r\Delta t + \sigma^2 j^2 \Delta t, \\ c_j &= -rj\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t \end{aligned}$$

mátrix elemek hasonló számolással megkaphatóak, mint az explicit módszerben a 3 egyenletnél. Az A mátrix $(n-2) \times n$ dimenziójú, tehát n ismeretlen van és $n-2$ egyenlet. A peremfeltételek miatt viszont f_i^0 és f_i^n minden i időpontban ismertek, így ez az alak a következőre redukálható:

$$\underbrace{\begin{bmatrix} b_{n-1} & c_{n-1} & 0 & \dots & & & 0 \\ a_{n-2} & b_{n-2} & c_{n-2} & \dots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_2 & b_2 & c_2 \\ 0 & 0 & 0 & \dots & 0 & a_1 & b_1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} f_i^{n-1} \\ f_i^{n-2} \\ \vdots \\ f_i^2 \\ f_i^1 \end{bmatrix}}_{\underline{f_i}} = \underbrace{\begin{bmatrix} f_{i+1}^{n-1} - a_{n-1} f_i^n \\ f_{i+1}^{n-2} \\ \vdots \\ f_{i+1}^2 \\ f_{i+1}^1 - c_1 f_i^0 \end{bmatrix}}_{\underline{f_{i+1}^*}}$$

Az A mátrix időhomogén, így az implementációban elegendő egyszer meghatározni az A LU-felbontását. Ezután az $A = LU$ felhasználásával, minden lépésben az $L\underline{y} = \underline{f_{i+1}^*}$ és $U\underline{x} = \underline{y}$ egyenletrendszerekből hatékonyan kiszámolható az $\underline{f_i} = \underline{x}$ opció árakat tartalmazó

vektor.

Az implicit módszer numerikus stabil, és a hibája $\mathcal{O}(\Delta t, (\Delta S)^2)$ nagyságrendű. [9]

2.4.3. Crank-Nicolson módszer

A Crank-Nicolson módszer az explicit és az implicit véges differencia módszerek kombinációja. Az 1 egyenletben szereplő parciális deriváltak Crank-Nicolson féle közelítése a következő alakban írható fel:

$$rf_i^j = \frac{f_{i+1}^j - f_i^j}{\Delta t} + \frac{1}{2} rj\Delta S \left[\frac{f_{i+1}^{j+1} - f_{i+1}^{j-1}}{2\Delta S} + \frac{f_i^{j+1} - f_i^{j-1}}{2\Delta S} \right] + \frac{1}{4}\sigma^2 j^2 (\Delta S)^2 \left[\frac{f_{i+1}^{j+1} - 2f_{i+1}^j + f_{i+1}^{j-1}}{(\Delta S)^2} + \frac{f_i^{j+1} - 2f_i^j + f_i^{j-1}}{(\Delta S)^2} \right],$$

amely lépésenként a 3 és a 4 egyenletek átlagolását jelenti. Viszont ez nem jelenti azt, hogy a Crank-Nicolson által adott megoldás a két módszer átlagával egyezik meg, mivel az explicit módszer lépései függenek az előző lépések megoldásától. A 4 egyenleteket az i időpontban minden j-re felírhatjuk lineáris egyenletrendszerként mátrix alakban is a peremfeltételek használatával:

$$A\underline{f_i} = B\underline{f_{i+1}} + \underline{c^*}$$

ahol

$$A = \begin{bmatrix} 1-b_{n-1} & -c_{n-1} & 0 & \vdots & & & 0 \\ -a_{n-2} & 1-b_{n-2} & -c_{n-2} & \vdots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & -a_2 & 1-b_2 & -c_2 \\ 0 & 0 & 0 & \vdots & 0 & -a_1 & 1-b_1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1+b_{n-1} & c_{n-1} & 0 & \dots & & & 0 \\ a_{n-2} & 1+b_{n-2} & c_{n-2} & \dots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_2 & 1+b_2 & c_2 \\ 0 & 0 & 0 & \dots & 0 & a_1 & 1+b_1 \end{bmatrix}$$

$$\underline{f_i} = \begin{bmatrix} f_i^{n-1} \\ f_i^{n-2} \\ \vdots \\ f_i^2 \\ f_i^1 \end{bmatrix}, \quad \underline{c^*} = \begin{bmatrix} a_{n-1}(f_i^n - f_{i+1}^n) \\ 0 \\ \vdots \\ 0 \\ c_{n-1}(f_i^0 - f_{i+1}^n) \end{bmatrix},$$

az A és B mátrix elemei pedig

$$\begin{aligned} a_j &= \frac{1}{4}\Delta t(rj + \sigma^2 j^2) \\ b_j &= -\frac{1}{2}\Delta t(r + \sigma^2 j^2) \\ c_j &= \frac{1}{4}\Delta t(-rj + \sigma^2 j^2). \end{aligned}$$

Az A és B itt is időhomogén mátrixok, így az explicit módszer implementációjához hasonlóan elég az első lépésben meghatározni az A mátrix inverzét vagy LU-felbontását. Ennek felhasználásával hátrafelé haladva $L\underline{y} = B\underline{f_{i+1}} + \underline{c^*}$ és $U\underline{x} = \underline{y}$ egyenletrendszerekből gyorsan számolható az $\underline{f_i} = \underline{x}$ vektor. Az Crank-Nicolson módszer numerikus stabil, és a hibája $\mathcal{O}((\Delta t)^2, (\Delta S)^2)$ nagyságrendű. [9]

2.4.4. ADI módszer

Az ADI *(Alternating Direction Implicit)* módszert Karel in 't Hout és Sven Foulon cikke alapján ismertetem, amelyben többek között különböző véges differencia módszereket hasonlítanak össze a Heston modellben.[10]

A Heston modell esetén az árazásnál a 2 differenciálegyenletet akarjuk numerikusan megoldani, a megfelelő peremfeltételek mellett. A BSM modellel szemben az árazásra használt rács a sztochasztikus varianciafolyamat miatt három dimenziósra nő. A rácspontok halmaza legyen

$$\{0, \Delta t, 2\Delta t, \ldots, l\Delta t = T\} \times \{0 = v_0, v_1, v_2, \ldots, v_m = V\} \times \{0 = s_0, s_1, s_2, \ldots, s_n = S\},$$

és az (t_i, v_j, s_k) rácsponton a derivatíva értékét jelöljük $f(t_i, v_j, s_k) = f_j^k(t_i)$-vel.

Rögzített t_i pontban az árakból álló két dimenziós rácsot reprezentáljuk a $G(t_i)$ mátrixszal, ahol $G(t_i)_{j,k} = f_j^k(t_i)$. Továbbá a mátrixokon definiáljuk egy indexelést, amely szerint tetszőleges $\mathcal{I} \in \mathcal{P}(\{0, 1, \ldots m\}) \setminus \{\varnothing\}$ és $\mathcal{J} \in \mathcal{P}(\{0, 1, \ldots n\}) \setminus \{\varnothing\}$ indexhalmazokra $G(t_i)[\mathcal{I}, \mathcal{J}]$ jelölje $G(t_i)$ azon részmátrixát, amely $G(t_i)$ $\mathcal{I}$-beli indexű sorainak $\mathcal{J}$-beli indexű oszlopelemeit tartalmazza. Azon indexeléseket, amelyeknél az összes sor- vagy oszlopindex szerepel, a teljes indexhalmaz kiírása helyett a : rövidebb jelölést használom (pl.: $G(t_i) = G(t_i)[:, :]$).

Az idő-tengelyen kívül a felosztás a térváltozók tengelye mentén nem feltétlen ekvidisztáns, melyet az indokol, hogy a lineáris egyenletrendszer mérete a felosztások lépésszámainak szorzatával $(n \cdot m)$ növekszik. A lényegesebb pontok környékének sűrítésével így lokálisan nagyobb pontosság érhető el alacsonyabb lépésszám esetén is.

A nem egyenletesen felosztott rácsot a [10] cikk alapján definiáltam. Az alaptermék árfolyamának tengelyén a kitüntetett pont amely körül sűrűbb felosztást akarunk felvenni legyen a kötési árfolyam K. A felosztás lépésszáma legyen n, és tekintsük az alábbi egyenletes ξ felosztást

$$\xi_k = \sinh^{-1}\left(\frac{K}{c}\right) + k\Delta\xi$$

ahol

$$\Delta\xi = \frac{\sinh^{-1}\left(\frac{S-K}{c}\right) - \sinh^{-1}\left(\frac{-K}{c}\right)}{n}.$$

Az alaptermék árfolyamának tengelyén a rácspontokat definiálja a ξ felosztás alábbi transz-

formáltja

$$s_k := K + c\sinh(\xi_k), \quad j = 0, \ldots, n.$$

Hasonlóan a sztochasztikus variancia tengelyén, a kitüntetett pont legyen a 0, a felosztás lépésszáma legyen m és tekintsük az η egyenletes felosztást

$$\eta_j = j\Delta\eta$$

ahol

$$\Delta\eta = \frac{\sinh^{-1}\left(\frac{V}{d}\right)}{m}.$$

A sztochasztikus varianca tengelyén a rácspontokat definiálja a η, következő transzformáltja

$$v_j := d\sinh(\eta_j) \quad k = 0, \ldots, m.$$

A képletekben szereplő c és d paraméterek segítségével állítható a kitüntetett pontok körüli felosztás sűrűsége, azaz a pontok hányad része essen az adott pont környékére. A K és a $v_0 = 0$ pontok körül közelítőleg igaz, hogy $s_{k+1} - s_k \approx c\Delta\xi$ és $v_{j+1} - v_j \approx d\Delta\eta$. Az implementáció során a [10] cikkhez hasonlóan $c = \frac{K}{5}$ és $d = \frac{V}{500}$ választást alkalmaztam.

A 2 parciális differenciálegyenletet diszkretizálása analóg az implicit és a Crank-Nicolson módszernél bemutatottaknál, viszont a nem ekvidisztáns felosztás miatt a jelölés a parciális deriváltak közelítésénél megváltozik. Az adott rácspont helyétől függően az elsőrendű parciális deriváltakat a térváltozók szerint háromféleképpen is közelíthetjük a (t_i, v_j, s_k) pontban. Legyen x egy felosztás és $\Delta x_i = x_i - x_{i-1}$, és definiáljuk a $J(x, i)$ mátrixot a következőképpen

$$J(x,i) =$$

$$\begin{bmatrix} \frac{\Delta x_i}{\Delta x_{i-1}(\Delta x_{i-1}+\Delta x_i)} & \frac{-\Delta x_{i-1}-\Delta x_i}{\Delta x_{i-1}\Delta x_i} & \frac{\Delta x_{i-1}+2\Delta x_i}{\Delta x_i(\Delta x_{i-1}+\Delta x_i)} & 0 & 0 \\ 0 & \frac{-\Delta x_{i+1}}{\Delta x_i(\Delta x_i+\Delta x_{i+1})} & \frac{\Delta x_{i+1}-\Delta x_i}{\Delta x_i\Delta x_{i+1}} & \frac{\Delta x_i}{\Delta x_{i+1}(\Delta x_i+\Delta x_{i+1})} & 0 \\ 0 & 0 & \frac{-2\Delta x_{i+1}-\Delta x_{i+2}}{\Delta x_{i+1}(\Delta x_{i+1}+\Delta x_{i+2})} & \frac{\Delta x_{i+1}+\Delta x_{i+2}}{\Delta x_{i+1}\Delta x_{i+2}} & \frac{-\Delta x_{i+1}}{\Delta x_{i+2}(\Delta x_{i+1}+\Delta x_{i+2})} \end{bmatrix},$$

valamint legyen

$$\underline{u_s}(t) = G(t)[j, \{k-2, k-1, ..., k+2\}]^T, \text{ és } \quad \underline{u_v}(t) = G(t)[\{j-2, j-1, ..., j+2\}, k].$$

Ekkor az alaptermék árfolyámának változójában vett elsőrendű parciális derivált a (t, x_i, v_k) pontban közelíthető a

$$\frac{\partial f}{\partial s}(t, v_j, s_k) \approx J(s,k)\underline{u_s}(t)$$

vektor egyik elemével, hasonlóan a variancia változóban

$$\frac{\partial f}{\partial v}(t, v_j, s_k) \approx J(v,j)\underline{u_v}(t).$$

A vektor középső elemét centrális differenciaként is nevezik és a $G(t)$ rács minden belső pontjában ezzel közelítem az elsőrendű parciális deriváltakat, kivéve azon pontokon, melyekre $v > 1$. Ezekben a pontokban a vektor első elemével, azaz a bal oldali differenciával közelítek, amellyel enyhíthető a módszer oszcillációja. [10] A rács szélein a bal vagy jobb oldali differenciával közelítem a deriváltakat úgy, hogy mindhárom közelítésben szereplő pont legyen rajta a rácson.

Másodrendű parciális deriváltakat centrális differencia segítségével közelítem a

$$\frac{\partial^2 f}{\partial s^2} =< \delta(s,k), \underline{u_s} >$$

$$\frac{\partial^2 f}{\partial v^2} =< \delta(v,j), \underline{u_v} >$$

alakban, ahol

$$\delta(x,i) = \left[\frac{2}{\Delta x_i(\Delta x_i + \Delta x_{i+1})} \quad \frac{-2}{\Delta x_i \Delta x_{i+1}} \quad \frac{2}{\Delta x_{i+1}(\Delta x_i + \Delta x_{i+1})}\right]^T,$$

és

$$\underline{u_s} = G(t)[j, \{k-1, k, k+1\}]^T, \quad \underline{u_v} = G(t)[\{j-1, j, j+1\}, k].$$

A vegyes parciális deriváltakat az elsőrendű deriváltaknál definiált J mátrix segítségével

$$\frac{\partial^2 f}{\partial s \partial v} = J(v,j)[2,:]^T G(t)[\{j-1,j,j+1\},\{k-1,k,k+1\}]J(s,k)[2,:]$$

alakban közelítem.

Peremfeltételeket a rácspontokat burkoló téglatest 6 lapjából 5-re szükséges megadni, hogy a feladat jól definiált legyen.

A fenti közelítéseket és a derivatíva által indukált peremfeltételeket felhasználva az implicit és a Crank-Nicolson módszerrel analóg módon a feladat az

$$\frac{\partial \hat{G}}{\partial t} = A\hat{G}(t) + b(t)$$

lineáris egyenletrendszer alakra hozható, ahol az A együtthatómátrix $(m-2)(n-2) \times (m-2)(n-2)$, a $b(t)$ vektor $(m-2)(n-2)$ dimenziójú és $\hat{G}(t)$ azt a vektort jelöli, amely $G(t)[\{2,\ldots,m\},\{2,\ldots,n\}]$ sorainak egymás mellé írásával, transzponálás után keletkezik.

A Crank-Nicolson módszer ebben az esetben már lesz nem eléggé hatékony, ugyanis az A mátrix bár időhomogén, de tridiagonális helyett $\min(m,n)$-től függő sávos mátrix. Az ADI módszer a többdimenziós feladatot az egyes tengelyek szerinti implicit lépésekre bontja fel, ahol már alkalmazható a Crank-Nicolson vagy az implicit módszer. A részfeladatok meghatározásához az A együtthatómátrixot és a $b(t)$ vektort bontsuk fel a vegyes deriváltak, az alaptermék árfolyam és a sztochasztikus variancia szerinti parciális deriváltakból kapott együtthatók szerint

$$A = A_0 + A_1 + A_2$$

és

$$b(t) = b_0(t) + b_1(t) + b_2(t)$$

alakban. Továbbá az $f_j^k(t)$ tagból származó r_d együtthatót osszuk szét az A_1 és A_2 között egyenlően.

Az A_1 és A_2 mátrixok által meghatározott lineáris egyenletrendszer ekkor átrendezhető a parcális deriváltak számolási módjától függően legfeljebb pentadiagonális alakra. Ekkor ezen időhomogén mátrixok sávszélessége fix, és nem függ $\min(m,n)$-től, ezáltal megoldhatóak

$\mathcal{O}(nm)$ lépésben.[10]

A paricális differenciálegyenletet a $t = T$-ból visszafelé haladva a 2.1 Craig-Sneyd séma (CS-ADI) szerint közelítve határozom meg az árakat. A Craig-Sneyd séma numerikusan feltétel nélkül stabil, és a hibája $\mathcal{O}((\Delta t)^2, (\Delta v)^2, (\Delta s)^2)$ nagyságrendű. [10]

2.1. Algoritmus Craig-Sneyd séma

Input: A, A_0, A_1, A_2: együtthatómátrixok
Input: $\hat{G}$: árazási rács
Input: $t_0, \ldots, t_n$: a rács időtengelyének felosztása
for $i = n - 1$ **to** 1 **do**
    $Y_0 = \hat{G}(t_{i+1}) + \Delta t \left(A\hat{G}(t_{i+1}) + b(t_{i+1}) \right)$
    $Y_1 = Y_0 + \frac{1}{2}\Delta t \left[\left(A_1 Y_1 + b_1(t_i) \right) - \left(A_1 \hat{G}(t_{i-1}) + b_1(t_{i-1}) \right) \right]$
    $Y_2 = Y_1 + \frac{1}{2}\Delta t \left[\left(A_2 Y_2 + b_2(t_i) \right) - \left(A_2 \hat{G}(t_{i-1}) + b_2(t_{i-1}) \right) \right]$
    $\tilde{Y}_0 = Y_0 + \frac{1}{2}\Delta t \left[\left(A_0 Y_2 + b_0(t_i) \right) - \left(A_0 \hat{G}(t_{i-1}) + b_0(t_{i-1}) \right) \right]$
    $\tilde{Y}_1 = \tilde{Y}_0 + \frac{1}{2}\Delta t \left[\left(A_1 \tilde{Y}_1 + b_1(t_i) \right) - \left(A_1 \hat{G}(t_{i-1}) + b_1(t_{i-1}) \right) \right]$
    $\tilde{Y}_2 = \tilde{Y}_1 + \frac{1}{2}\Delta t \left[\left(A_2 \tilde{Y}_2 + b_2(t_i) \right) - \left(A_2 \hat{G}(t_{i-1}) + b_2(t_{i-1}) \right) \right]$
    $\hat{G}(t_i) = \tilde{Y}_2$.
end for
return $\hat{G}$

2.5. Implementáció

A Crank-Nicolson és a CS-ADI módszereket Python 3.12-ben implementáltam a *numpy* és *scipy* könyvtárak segítségével. A dolgozathoz készített kódbázis és a felhasznált könyvtárak és csomagok verziószámmal megtalálhatóak egy nyilvános *Github repository*-ban:

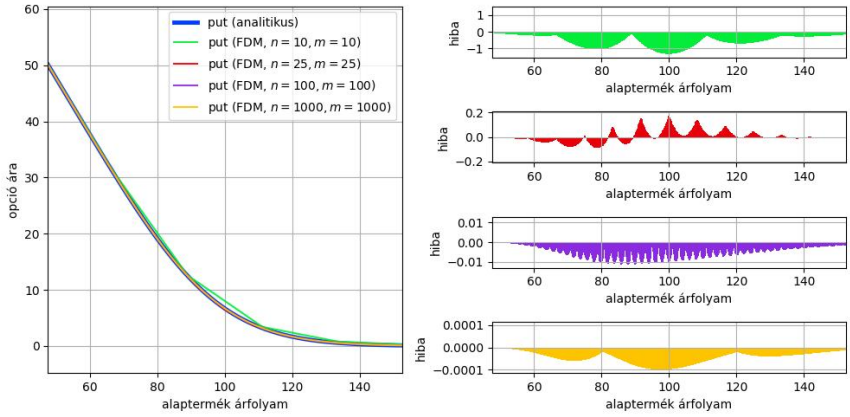
`https://github.com/HKristof136/msc_thesis_2025`.

A Crank-Nicolson módszernél a peremfeltételeket a lejáratkor és az alaptermék árfolyamának tengelyén a két szelső pontra szükséges megadni. A 2 táblázat összegzi a BSM modellben a Crank-Nicolson módszernél implementált peremfeltételeket.

Opció típusa	$t = T, \forall s$	$\forall t, s = S_{\min}$	$\forall t, s = S_{\max}$
c_K	$\max(s - K, 0)$	0	$S_{\min} - e^{-rt}K$
p_K	$\max(K - s, 0)$	$e^{-rt}K - S_{\min}$	0
$c_{K,B}^{UO}$	$\mathbb{I}_{s<B} \max(s - K, 0)$	0	0
$p_{K,B}^{UI}$	$\mathbb{I}_{s>B} \max(K - s, 0)$	0	$p_K(S_{\max})$
$c_{K,B}^{DI}$	$\mathbb{I}_{s<B} \max(s - K, 0)$	$c_K(S_{\min})$	0
$p_{K,B}^{DO}$	$\mathbb{I}_{s>B} \max(K - s, 0)$	0	0

2. táblázat. Peremfeltételek a Crank-Nicolson implementációban

Az árazás a rácson belül tetszőleges alaptermék $(S_0, T - t)$ pontban lehetséges interpolációval. Az implementációban kihasználva a tengelyenkénti egységes felosztást, bilineáris interpolációt alkalmazok, így a nemlineáris függvények közelíse miatt az interpolációs hiba a teljes hiba nagy részéért felelős. Az 5 ábrán a Crank-Nicolson módszerből számolt ár és az analitikus Black-Scholes formula közötti hiba látható egy európai put opció esetén.



5. ábra. Black-Scholes formula közelítése a Crank-Nicolson módszerrel egy európai put opcióra az alaptermék árfolyamának függvényében. $S \in [50, 150], K = 100, r = 10\%, \sigma = 40\%, T = \frac{90}{365}$.

Az 3 táblázatokban a Crank-Nicolson módszer átlagos négyzetes hibája és a másodpercben mért átlagos futási idők láthatóak különböző felosztásszámokra.

		m				
		5	**10**	**25**	**50**	**100**
n	**5**	1,277664	1,277855	1,277895	1,277900	1,277901
	10	0,224546	0,224373	0,224338	0,224333	0,224332
	25	0,001225	0,001406	0,001450	0,001455	0,001457
	50	0,000328	0,000260	0,000260	0,000261	0,000261
	100	0,000560	0,000019	0,000016	0,000016	0,000016

		m				
		5	**50**	**100**	**500**	**1000**
n	**5**	0,0102	0,0293	0,0374	0,1485	0,2996
	50	0,0061	0,0212	0,0414	0,1860	0,3258
	100	0,0086	0,0256	0,0424	0,1775	0,3640
	500	0,3325	0,2833	0,4861	1,8814	3,5432
	1000	0,5077	1,2648	2,0857	7,3981	14,5321

3. táblázat. Crank-Nicolson módszer átlagos négyzetes hibája, valamint átlagos futási ideje másodpercben európai put opcióra, különböző rácsméretek esetén. $S \in [1, 200], K = 100, r = 5\%, \sigma = 20\%, T = 1$.

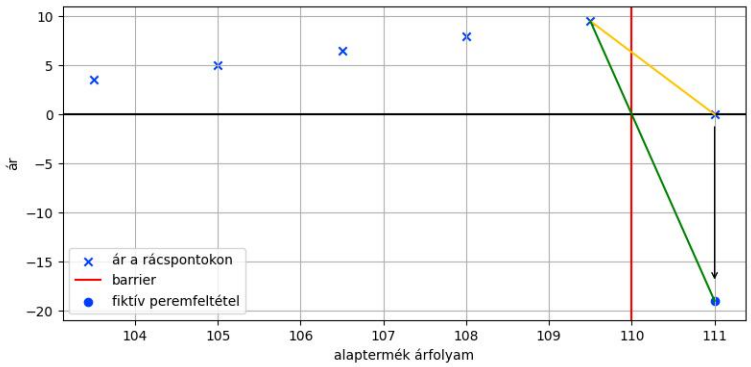
A barrier opciók kifizetésfüggvénye nem folytonos az alaptermék árfolyamának függvényében. Parciális differenciálegyenleteknél a numerikus módszerek nem folytonos peremfeltételek esetében hajlamosak az oszcillációra. Barrier opcióknál az árazáshoz használt rács leredukálódik az alaptermék árfolyamának tengelyén, ugyanis típustól függően elegendő csak a B-ig vagy a B-től meghatározni a rácsot. Ekkor hasznos az alaptermék árfolyamának tengelyén a felosztást úgy megválasztani, hogy a barrier (B) két rácspont között legyen. A kifizetésfüggvény az árazási rácson ekkor egyetlen pontpár közötti szakasz kivételével folytonos.

Minden lépésben ennél a pontpárnál a rács szélére eső pontban a peremfeltételt módosítsuk egy fiktív értékre úgy, hogy rajta legyen a szomszédos pont által meghatározott és a

B pontban ismert termékspecifikus árak által meghatározott egyenesen. Azaz

$$\hat{f}^l_{j+1} = \frac{f(B) - f^k_{j+1}}{B - S_k}(S_l - S_k) + f^k_{j+1}, \tag{5}$$

ahol l a szélső pont indexe, k pedig a vele szomszédos rácspont. Az 6 ábrán a módosított peremfeltétel hatása látható egy európai KO call opcióra. A numerikus módszer így kevésbe oszcillál B környezetében és a hiba nagyságrendje továbbra is $\mathcal{O}((\Delta t)^2, (\Delta S)^2)$ marad a Crank-Nicolson módszernél [9].



6. ábra. Fiktív pontok hatása a parciális deriváltra egy "up-and-out" barrier call opció esetén az első lépésben, $B = 110, K = 100, r = 0.04, \sigma = 0.1$.

A CS-ADI módszert ritka mátrixok segítségével implementáltam szintén Python-ban. Az árazó rács $100 \times 100 \times 200$-as felosztással 32-bites lebegőpontos számokat használva ≈ 8 Mb tárhelyet foglal. A lineáris egyenletrendszerek együtthatómátrixai viszont ekkor $100 \cdot 200 \times 100 \cdot 200$ dimenziójúak, amelyek egyenként ≈ 1.5 Gb memóriát igényelnének. Ritka mátrixok használatával a memóriaigény lényegesen csökkenthető, hiszen a mátrixok legfeljebb pentadiagonálisak, ezért a nem nulla elemek száma $\mathcal{O}(n \cdot m)$. Ezáltal a 4.2.1 alfejezetben ismertetett adatgenerálás során az árazás akár több processzormagon is futtatható párhuzamosan.

Árazáskor a felosztás növelésével a lineáris egyenletrendszer mérete négyzetesen növekszik, amely a nem hardverközeli programozási nyelveknél - mint a Python - jelentős futásidő növekedést okoz. A 4 táblázatban megfigyelhető, hogy a futásidő már kisebb felosztásméretek esetén is lényegesen nagyobb a Crank-Nicolson módszer azonos felosztásmérethez tartozó futásidejénél.

		n				
		5	**25**	**50**	**100**	**250**
m	**5**	0,2516	0,2681	0,2845	0,3346	0,4054
	25	0,2703	0,3442	0,4379	0,5944	1,1262
	50	0,3147	0,4375	0,6010	0,9658	2,1332
	100	0,3430	0,6402	1,0444	1,7887	5,3975
	250	0,5897	1,6534	2,8171	5,5328	15,2637

4. táblázat. ADI módszer átlagos futási ideje másodpercben európai call opcióra, különböző alaptermék árfolyam és sztochasztikus variancia tengelyfelosztás-méretek esetén. Az időtengely felosztásának mérete mindegyik esetben 100.

Az implementációban a [10] cikkben is használt Neumann és Dirichlet peremfeltételeket alkalmaztam, amelyek egy K kötési árfolyamú európai call opció esetén,

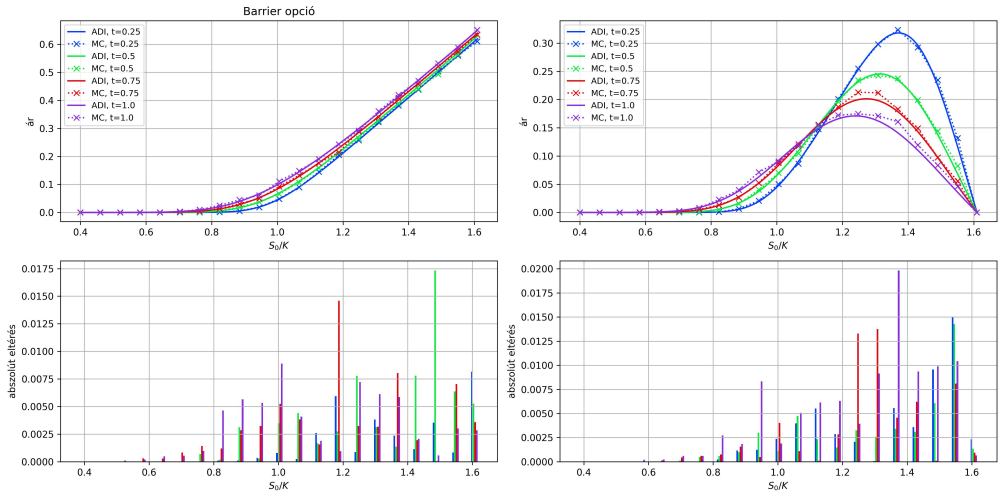
$$\begin{aligned} f_j^k(T) &= \max(0, s_k - K) \quad \forall j, k, \\ f_j^0(t_i) &= 0 \quad \forall j, i, \\ \frac{\partial f}{\partial s}(t_i, v_j, S) &= 1 \quad \forall j, i, \\ f_m^k(t_i) &= \max(0, s_k - e^{-rt}K) \quad \forall k, i. \end{aligned}$$

A $v = 0$ esetnél nem szükséges peremfeltételt megadni, ugyanis a 2.1 Feller-feltétel biztosítja, hogy a sztochasztikus varianca folyamat a megoldás során szigorúan pozitív marad.

A KO barrier call opció esetén, amely "*up-and-out*" típusú B barrierrel és K kötési árfolyammal, a peremfeltételek az európai call opcióhoz képest három helyen változnak

$$\begin{aligned} f_j^n &= 0 \quad \forall j, i, \\ f_m^k(t_i) &= 0 \quad \forall k, i \quad , \end{aligned}$$

valamint a B pont környékén ugyanúgy a 5 egyenlet adja meg a parciális derivált értékét a Neumann-féle peremfeltételhez. A 7 ábrán a CS-ADI és a Milstein módszer szerinti opcióárak összehasonlítása szerepel call, valamint KO barrier opciók esetén különböző lejáratokra.



7. ábra. Monte Carlo, valamint a CS-ADI módszer által számolt árak összehasonlítása a Heston modellben. $\kappa = 2.0, \theta = 0.04, \sigma = 0.2, \rho = -0.3, V_0 = 0.04, B = 1.6$

A görög betűk meghatározáshoz az árazó rács tengelyváltozói esetében numerikus közelítést alkalmazok, az interpoláció használatával. Azon változók esetén amelyek nem vesznek részt a rács definiálásában, a numerikus közelítésnél szükséges újraszámolni az árakat a rácson. Így például a Crank-Nicolson módszernél az összes görög betű meghatározásához a differenciálegyenletet öt rácson kell megoldani.

3. Neurális hálók

Ebben a fejezetben a neurális hálók matematikai leírását ismertetem *Ian Goodfellow, Yoshua Bengio* és *Aaron Courville Deep Learning* című könyve alapján.[11]

3.1. Előrecsatolt neurális hálók

A legelterjedtebb változata a neurális hálóknak az előrecsatolt hálók, melyek célja egy legtöbbször nem ismert $f^\star$ függvény lehető legpontosabb közelítése, általában címkézett adathalmazon.

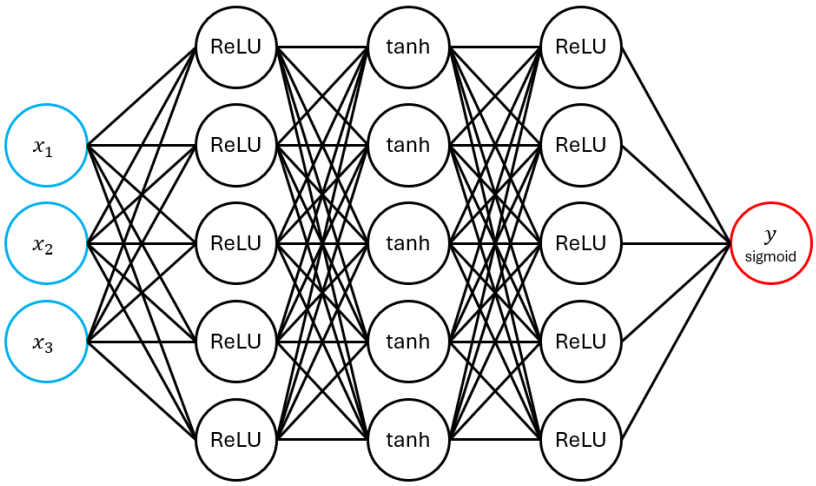
$$\underline{y} = f^\star(\underline{x})$$

Az előrecsatolt hálók építőkövei a mesterséges neuronok, melyeket reprezentálhatunk tetszőleges $g : \mathbb{R}^k \to \mathbb{R}$ függvényként. Általában a g függvény a lineáris regressziónál is használt affin transzformáció és egy nemlineáris aktivációs függvény kompozíciója

$$g(\underline{x}, \theta) = h(w^T \underline{x} + b),$$

ahol w a neuron súlyvektora, b pedig az eltolás paramétere. Előrecsatolt hálóknál a neuronok rétegekben helyezkednek el, és általában az azonos rétegben elhelyezkedő neuron inputjai ugyanazok. A háló végső rétege az output réteg, amely kapcsolódhat közvetlenül az inputhoz is, ekkor a halót egyrétegűnek nevezzük. A mély neuron hálózatok, olyan neurális hálók melyeknek több neuron rétegük is van, a rétegek száma a háló mélysége, a rétegekben található neuronok számának maximuma pedig a háló szélessége. Egy neurális háló paraméterein (θ) a neuronjai affin transzformációjában szereplő súlyvektorok és eltolás paraméterek összességét értjük.

Legyen a neuronháló mélyége d, szélessége k és az aktivációs függvény az i-edig rétegben $g_i(\underline{x})$, ekkor a háló reprezentálható, mint $F(\underline{x}) = (g_k \circ g_{k-1} \circ \cdots \circ g_1)(\underline{x})$ függvény. Az 8 ábrán egy 3 mélységű és 5 szélességű teljes háló kapcsolódási struktúrája szerepel.



8. ábra. Teljes mély neurális háló kapcsolódási struktúrája.

A neurális hálók kezdeti alkalmazásainál aktivációs függvényekként gyakran a *szigmoid* $\sigma(x) = \frac{1}{1+e^{-x}}$ és *tanh* függvényeket használtak. A probléma ezekkel a függvényekkel, hogy a nullától távoli argumentumokra a deriváltjuk közel 0, ami az eltűnő gradienshez vezet és ezzel megnehezíti a háló tanítását.

Modern alkalmazásoknál a legelterjedtebb aktivációs függvény a *ReLU (rectified linear unit)*, ahol a

$$h(z) = max\left\{0, z\right\},$$
$$g(\underline{x}, \theta) = h(w^T \underline{x} + b).$$

A *ReLU*előnye, hogy a teljes aktivációs tartományában megegyezik az identitás függvénnyel. Ezáltal a gradiens nem tűnik el nagy értékeknél, és a linearitás miatt konzisztensen 1 az értéke. Kiértékelése is sokkal hatékonyabb, mint az előző aktivációs függvényeké, hiszen nem kell az exponenciális függvény Taylor-sorát kiszámolni. Hátránya viszont, hogy gradienssel alapú optimalizálás során a nem aktív neuronok nem képesek a tanulásra, ami a halott *ReLU*neuronok problémához vezet, amelynél a háló akár nagy része a tanítás során visszafordíthatatlan módon konstans 0-vá válik.

A *ReLU*, tanh és szigmoid aktivációs függvényekre igaz az alábbi tétel, amely elméleti hátteret biztosít folytonos függvények neuronhálókkal való közelítésének. [12]

3.1. Definíció. *Univerzális approximátor tulajdonság*
Legyen Φ neurális hálók egy családja, Φ rendelkezik az univerzális approximátor tulajdonsággal egy $\boldsymbol{f}$ függvénytérben, ha $\forall f \in \boldsymbol{f}$-re $\exists(\phi)_n$ neuronhálókból álló sorozat melyre

$$\lim_{n\to\infty} \phi_n = f.$$

3.1. Tétel. *Legyen $\Phi = \{\phi : \phi$ neurális háló melynek minden neuronjának aktivációs függvénye ReLU, szigmoid vagy* tanh*$\}$, ekkor Φ rendelkezik az univerzális approximátor tulajdonsággal a folytonos függvények terében.*

A *ReLU*függvénynek az elterjedése óta több továbbfejlesztése is megjelent, ilyen például az *átengedő-ReLU (leaky-ReLU)*, amely a *ReLU* függvény nem aktív értelmezési tartományához is nem nulla gradiens rendel:

$$g(\underline{x}, \theta, \alpha) = max\left\{0, w^T\underline{x} + b\right\} + \alpha min\left\{0, w^T\underline{x} + b\right\}.$$

A *Softplus* és a *GELU* aktivációs függvények a *ReLU*minden pontban differenciálható közelítései. A Softplus nemnegatív és a függvénye megadható

$$g(\underline{x}, \theta, \beta) = \frac{1}{\beta}\log\left(1 + \exp(\beta(w^T\underline{x} + b)\right),$$

míg a GELU esetén

$$g(\underline{x}, \theta) = \frac{1}{2}\left(w^T\underline{x} + b\right)\left[1 + \tanh\left(\sqrt{\frac{2}{\pi}}(w^T\underline{x} + b + 0.044715(w^T\underline{x} + b)^3\right)\right]$$

alakban. Az átengedés és a sima közelítés esetén a neuronok nem tudnak a tanítás során véglegesen deaktiválódni. Empirikusan viszont nem érnek el lényegesen jobb eredményeket, mint a *ReLU*. [11]

A neurális hálók legjelentősebb célja olyan függvények közelítése a rendelkezése álló adatokból, amelyek általában egyáltalán nem ismeretek. Egy másik alkalmazási terület olyan függvények közelítése, amelyeket nem tudunk gyorsan kiszámolni. Neurális hálóknál - a struktúrájukból adódóan - a kiértékelés hatékony, így egy már tanított háló a komplexitást

le tudja csökkenteni $\mathcal{O}$(háló mérete)-re. Derivatív termékek árazásánál ez a tulajdonság a modellkalibráció során akár jelentős gyorsítást is eredményezhet [1].

A hálók tanításán a paraméterek optimalizálását értjük, amely során a háló paramétereit az optimalizáló algoritmus iteratívan változtatja egy veszteségfüggvény alapján. Ez az eljárás neurális hálók esetében gradiens alapú. A veszteségfüggvényt általában a probléma típusa indukálja, regressziós problémánál a leggyakoribb választás az átlagos négyzetes hiba veszteségfüggvény *(MSE)*,

$$MSE(F(X), \underline{y}) = \frac{1}{n} \sum_i \left(F(\underline{X_i}) - y_i \right)^2$$

ahol X egy n sorból álló mátrix, $F(X)$ a háló soronkénti kiértékeléseiből álló n dimenziós vektor.

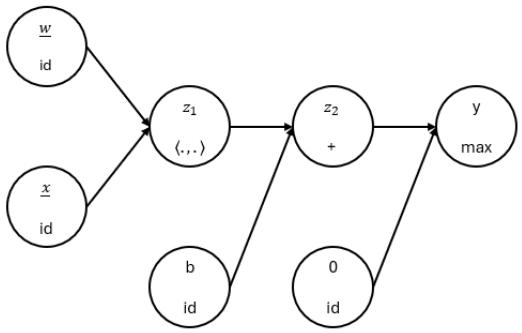
3.2. Automatikus differenciálás

A neurális hálók gradiensalapú módszerrel történő hatékony tanításához a gradiensek gyors számolására van szükség. Az automatikus differenciálás egy olyan kutatási terület, amely ezen parciális deriváltakat gyorsan, pontosan és hatékonyan meghatározó algoritmusokat fejleszt. Neurális hálóknál a leggyakrabban alkalmazott algoritmus a láncszabályon alapuló *backpropagation*. Az algoritmus alapgondolatát 1962-ben Frank Rosenblatt vetette fel, míg az első tetszőleges differenciálható függvények kompozíciójára is alkalmazható konstrukciót Seppo Linnainmaa publikálta 1970-ben. Az ezt követő évtizedekben az eljárás nem volt hatékonyabb más gépi tanulási módszereknél. Ez egészen addig így is maradt, amíg a GPU-technológia fejlődése a közelmúltban lehetővé nem tette a mély neuronhálók nagy adathalmazokon történő tanítását az algoritmus segítségével.

Az algoritmust $\nabla_x f(\underline{x}, \underline{y})$ parciális deriváltak számolásához fejlesztették ki tetszőleges f függvényre, ahol $\underline{x}$ az f bemeneti változóinak azon halmaza, amelyek szerint szükség van a parciális deriváltakra.

Az eljárás elvégzéséhez szükséges az f számítási gráfjának a felépítése. A számítási gráf pontjait számítási egységekként reprezentálhatjuk, amelyek mindegyikéhez hozzá van rendelve egy egyszerű függvény az engedélyezett alapműveletek halmazából. Ezen alapműveletek

halmaza olyan függvényekből áll amelyeknek deriválása $\mathcal{O}(1)$ időben elvégezhető.



9. ábra. $ReLU$aktivációs függvény számítási gráfja.

Egy számítási egység az inputjaira kiértékeli a függvényt és a kimenete inputként szolgálhat más számítási egységeknek. A számítási gráfban legyen egy i-ből a j egységbe mutató irányított él, ha az i szükséges a j egység kiszámításához inputként. Az f számítási gráfjában csak a bemeneti változókhoz tartozó egységeknek és egyéb konstansoknak nincsen szülője, valamint csak egy egységnek nincsen leszármazottja, ennek az egységnek a kimeneti értéke az $f(\underline{x}, \underline{y})$. Az 9 ábrán a $ReLU$aktivációs függvény számítási gráfja szerepel, amely gyakori választás egy előrecsatolt neurális háló neuronjainak. Az irodalomban gyakran a neuronokat, vagy akár a teljes rétegeket értik számítási egységek alatt, hogy a bonyolultabb neurális hálók struktúráját könnyebb lehessen reprezentálni.

Az algoritmus két fáziusú a számítási gráf felépítése után. Az 3.1 előre fázisban a szükséges információ az irányított élek mentén az egységeken keresztül végigáramlik a számítási gráfon, ezáltal kiértékelve az f függvényt az $\underline{x}, \underline{y}$ helyen.

3.1. Algoritmus Backpropagation első fázis

Input: $\mathcal{G}$: az f számítási gráfja n számítási egységgel
Input: $\mathcal{B}$: az egységek bejárási sorrendje, ahol az első k egység input változó és az n-edik a kimeneti egység
for $i = k + 1$ **to** n **do**
    $v^i \leftarrow \mathcal{G}[\mathcal{B}(i)]$
    $\mathcal{A}^i \leftarrow \{\mathcal{G}[\mathcal{B}(j)] : \exists(\mathcal{G}[\mathcal{B}(j)], v^i)$ él a $\mathcal{G}$ gráfban$\}$
    $v^i \leftarrow f^i(\mathcal{A}^i)$
end for
return $\mathcal{G}[\mathcal{B}(n)]$

A 3.2 hátra fázisban az algoritmus a kimeneti egységtől kezdve, a szülők felé haladva a gráfon kiszámolja a parciális derivált értékeket az élek mentén a láncszabályt iterálva. Ennek a fázisnak kétféle implementációja is létezik, attól függően, hogy a futásidő vagy a felhasznált memória csökkentése a cél.

3.2. Algoritmus Backpropagation második fázis

1: **Input:** $\mathcal{G}$, az f számítási gráfja n számítási egységgel
2: **Input:** $\mathcal{B}$, az egységek egy bejárási sorrendje
3: **Inicializálás:** *grad_table*: egy adatstruktúra, amely az egységekhez tartozó parciális deriváltakat tárolja
4: **for** $j = n - 1$ **to** 1 **do**
5: $v^j \leftarrow \mathcal{G}[\mathcal{B}(j)]$
6: $\mathcal{D}^j \leftarrow \{\mathcal{G}[\mathcal{B}(i)] : \exists(v^j, \mathcal{G}[\mathcal{B}(i)])$ él a $\mathcal{G}$ gráfban$\}$
7: $grad\_table[v^j] \leftarrow \sum_{v^i \in \mathcal{D}^j} grad\_table[v^i] \cdot \frac{\partial v^i}{\partial v^j}$
8: **end for**
9: **return** $\{(i, grad\_table[\mathcal{B}(\mathcal{G}, i)]) : i = 1, \ldots, k\}$

A futásidő javítható, ha a köztes gradiens értékeket egy megfelelő adatstruktúrában - például egy hash táblában - tároljuk el, melyeket az iteráció során $\mathcal{O}(1)$ időben el tudunk érni a leszármazottakra. A felhasznált memória lecsökkenthető, ha nem minden egységre tároljuk el a gradiens értékeket. A számítási gráf bizonyos részgráfjain érdemes lehet minden parciális deriváltat többször is kiszámolni, amellyel bár a futásidő növekszik, a szükséges tárhely lényegesen lecsökkenhet.

A backpropagation algoritmus lényegében a számítási gráf gradiens táblázatának hatékony feltöltése dinamikus programozás segítségével. Az algoritmus mindkét fázisa elvégezhető egy n csúcs számítási gráfon $\mathcal{O}(n)$ időben, ha a számítási egységek kiértékelése és deriválása is $\mathcal{O}(1)$ idejű. A modern neurális háló architektúra implementációkban a backpropagation algoritmus a számítási gráf felépítésében és a hátrafelé iteráció lépéseiben is több egyéb optimalizálást hajt végre.

Ha az f függvény egy neurális háló kimenete, akkor az algoritmus első fázisa szerint a kiértékelés elvégezhető $\mathcal{O}(d \cdot k)$ időben, ahol d a háló mélysége és k a háló szélessége k. A második fázishoz szükséges még egy számítási egységet hozzávenni a gráfhoz, amely háló kimenetéhez tartozó J veszteségfüggvényt számolja, majd a parciális deriváltakat eszerint számolni a háló paramétereinek optimalizálásához.

Pénzügyi termékek árazásánál az, hogy a kiértékelés ideje a neuronháló számítási gráfjának méretével arányos, több modellnél akár lényeges gyorsítást is eredményezhet. Továbbá az ár kiszámításának hatékonysága mellett, a backpropagation algoritmus az összes görög betűt is képes visszaadni ugyanazzal a komplexitással.

3.3. Neurális hálók tanítása

A neurális hálók tanítása során egy tanító algoritmus a θ paramétereket úgy változtatja, hogy a $J(\theta)$ veszteségfüggvény értéke minél kisebb legyen. Tehát az algoritmus próbál olyan $\theta^\star$ paramétereket keresni amely minimalizálja a

$$J(\theta) = \mathbb{E}_{(X,\underline{y})\sim P}\left[L(F(X \mid \theta), \underline{y})\right] \tag{6}$$

rizikó mennyiséget, ahol az X az adathalmaz amelyen tanítjuk a halót, y az X-hez tartozó igazi kimeneti értékek, P az adatgeneráló folyamat és L az egyes adatpontokra vonatkozó hibafüggvény.

Gépi tanulásnál az optimalizáció eltér a hagyományos módszerektől, hiszen megfigyelt tanító adatokból akarunk minél kisebb általánosítási hibát indirekten elérni. Azaz olyan paramétereket választani a neurális hálóhoz, amellyel minimalizáljuk a hibát olyan adatokra is amelyek nem vettek részt a háló tanításában. Általában a P egyáltalán nem ismert, vagy az adatok generálása költséges, és így a neurális hálóknál a rizikó helyett az optimalizáló algoritmusok az empirikus rizikót

$$\mathbb{E}_{(X,\underline{y})\sim \hat{P}}\left[L(F(X \mid \underline{y})\right] = \frac{1}{n}\sum_{i=1}^{n} L(\underline{X_i}, y_i)$$

minimalizálják feltételezve, hogy ezzel a általánosítási hiba is csökken. Nemlineáris aktivációs függvények használata esetén az minimalizálandó veszteségfüggvény nem konvex a paramétertéren. Ezáltal a gradiens módszerrel megtalált lokális minimum általában nem lesz automatikusan globális minimum, továbbá a tanulási folyamat megakadhat, ha az algoritmus nyeregpontot talál meg. Ennek ellenére a gyakorlatban mégis kizárólag gradiens alapú optimalizáló algoritmusok vannak használatban a neurális hálóknál. Az $\mathbb{R}^n \to \mathbb{R}$ függvények

esetében az n növelésével a nyeregpontok és lokális mimimum pontok aránya exponenciális sebességgel nő [13], és a lokális minimum pontok az alacsony veszteségfüggvény-értékek közelében sokkal sűrűbben fordulnak elő. Empirikusan megfigyelhető az alkalmazásoknál, hogy a lokális minimum pontoknál a veszteségfüggvényértéke már elfogadhatóan alacsony, és a sztochasztikus gradiens süllyedést használó algoritmusok a nyeregpontokon a legtöbb esetben könnyen túljutnak. [11, 8.2.3 alfejezet.]

A gradiens süllyedés módszernél az iteráció során a paramétereket a gradiens irányában változtatjuk, ezáltal az empirikus rizikó minden lépésben csökken. A gradiens értékeket a 3.2 alfejezetben bemutatott backpropagation algoritmus segítségével hatékonyan lehet számolni.

Gyakorlati alkalmazásoknál az összes adat figyelembevétele az empirikus rizikó kiszámításához túlságosan számításigényes. Ezért az algoritmusok sztochasztikusak abban az értelemben, hogy az adathalmaz csak egy véletlen mintáján számolják ki az empirikus rizikó gradiensét és a paramétereket ebben az irányban módosítják. A véletlen mintát *mini-batch*, vagy rövidebben *batch*-nek szokás nevezni. A batch-en számolt gradiens egy torzítatlan becslése annak a gradiensnek amit minden adat figyelmbevételével kapnánk. Mivel azonban a tanítás célja a 6 rizikó csökkentése - amelynél a tényleges gradienst nem ismerjük -, a teljes adathalmazon számolt gradiens ugyancsak egy torzítatlan közelítés lenne. A gradiensek átlagának közelítésekor a sztenderd hiba b batch méret esetén $\sigma/\sqrt{b}$, vagyis négyzetesen nagyobb batch méretre van szükség ahhoz, hogy a közelítés hibája ugyanakkora mértékben csökkenjen.

A neurális hálóknál empirikusan is megfigyelhető, hogy a tanulás hatékonyabb, ha paraméter változtatásokat kis batch méret mellett végezzük el. A modern GPU-kon a párhuzamosítás miatt a futásidő gyorsabb, ha a batch méretet 2 hatványnak választjuk. Emiatt a leggyakrabban használt batch méretek 16 és 2048 közötti intervallumban találhatóak.

A 3.3 sztochasztikus gradiens süllyedés algoritmus (SGD) és módosításai a leggyakrabban használt tanító algoritmusok. Az SGD algoritmusnál az adathalmazt batch-ekre osztjuk fel és az iteráció során a háló paramétereit a gradiens becslések irányában módosítjuk, az ϵ tanulási ráta mértékével. Egy teljes iterációt az adathalmazon *epoch*-nak szokás nevezni, az első epoch során a gradiensek még torzítatlan becslései 6 rizikó valódi gradiens értékeinek. Későbbi epoch-oknál, ha ugyanazt az adatpontot többször is felhasználunk ez a tulajdonság

már nem lesz igaz, de több epoch használata szignifikánsan csökkenti az empirikus rizikót.

3.3. Algoritmus Sztochasztikus gradiens süllyedés momentummal

Input: $(X, \underline{y})$: adathalmaz, F: neurális háló, θ: kezdeti paraméterek
Input: m: az epoch-ok száma, b: a batch méret
Input: α: a momentum exponenciális lecsengésének sebessége
Input: ϵ_i: a tanulási ráta az i-edik epoch-ban.
$k \leftarrow \lfloor n/b \rfloor$: a batch-ek száma
$\{(X^1, \underline{y^1}), \ldots, (X^k, \underline{y^k})\}$: a batch-ek halmaza
$v \leftarrow 0$: kezdeti momentum
for $i = 1$ **to** m **do**
    for $j = 1$ **to** k **do**
        $g \leftarrow \frac{1}{b} \sum_l \nabla_\theta L \left(F(\underline{X_l^j} \mid \theta), y_l^j \right)$
        $v \leftarrow \alpha v - \epsilon_i g$
        $\theta \leftarrow \theta + v$
    end for
end for

Az ϵ az SGD egyetlen, de fontos hiperparamétere, nem megfelelő megválasztása lassú konvergenciát vagy instabil tanulást eredményezhet. Gyakran az ϵ konstans helyett exponenciálisan lecsengő az epoch-ok mentén. Ezáltal a tanulási folyamat végére az algoritmus egyre kisebb lépésekben változtatja a paramétereket egy potenciális lokális minimumhely környékén.

Az 3.3 algoritmus az SGD egyik módosítása, amely momentumot használ a gradienseknél. A momentum az előző lépéseknél használt gradiensek exponenciálisan csökkenő mozgóátlaga. A momentum fel tudja gyorsítani a tanulási folyamat konvergenciáját, ha az empirikus rizikó gradiens becsléseinél nagy a szórás.

Az SGD algoritmus a neurális háló minden paraméterénél ugyanazzal az ϵ tanulási rátával változtat az iterációk során. Általában a veszteségfüggvény paraméterenként különböző mértékben érzékeny, ezért ugyanannak a tanulási rátának a használata lassabbá teheti a tanulást bizonyos tengelyek mentén. Ezt a problémát az adaptív tanulási rátát használó algoritmusok extra hiperparaméterek hozzáadása nélkül tudják kezelni.

3.4. Algoritmus Adam

Input: $(X, \underline{y})$: adathalmaz, F: neurális háló, θ: kezdeti paraméterek
Input: m: az epoch-ok száma, b: a batch méret
Input: β_1, β_2: a momentumok exponenciális lecsengésének sebessége
Input: ϵ_i: a tanulási ráta az i-edik epoch-ban.
$k \leftarrow \lfloor n/b \rfloor$: a batch-ek száma
$\{(X^1, \underline{y^1}), \ldots, (X^k, \underline{y^k})\}$: a batch-ek halmaza
$v_1 \leftarrow 0$: kezdeti momentum
$v_2 \leftarrow 0$: kezdeti második momentum $\delta \leftarrow 10^{-7}$: konstans a numerikus stabilitásért
for $i = 1$ **to** m **do**
    for $j = 1$ **to** k **do**
        $g \leftarrow \frac{1}{b} \sum_l \nabla_\theta L \left(F(\underline{X_l^j} \mid \theta), y_l^j \right)$
        $v_1 \leftarrow \beta_1 v_1 + (1 - \beta_1) g$
        $v_2 \leftarrow \beta_2 v_2 + (1 - \beta_2) g^T g$
        $\hat{v}_1 \leftarrow \frac{v_1}{1 - \beta_1^i}$
        $\hat{v}_2 \leftarrow \frac{v_2}{1 - \beta_2^i}$
        $\theta \leftarrow \theta - \epsilon_i \frac{\hat{v}_1}{\sqrt{\hat{v}_2} + \delta}$ ▷ a műveletek elemenként értendőek
    end for
end for

Az egyik legnépszerűbb adaptív tanulási rátával rendelkező tanítási algoritmus az 3.4 Adam (melynek neve az *adaptive moments* rövidítéséből származik). [14] Az Adam algoritmus a paramétereken különböző - a tanítási trajektória mentén dinamikusan változó - tanulási rátát alkalmaz, a momentum és a normanégyzet segítségével. Az első és a második (nem centrált) momentumoknál a kezdeti érték 0, ezért az algoritmus minden lépésben exponenciálisan lecsengő torzítás korrekciót is alkalmaz.

A tanító algoritmusok között nem ismert olyan, amely egyértelműen jobb lenne a többinél, ezért az algoritmust is szokás a neurális háló hiperparamétereként kezelni.

A kezdeti θ paraméterek megválasztására több heurisztikus eljárás is létezik. Az egyik fontos tulajdonság a paraméterek megválasztásánál, hogy ugyanazokat az inputokat használó számítási egységeknek legyen különböző paraméterük. Ugyanis szimmetria esetén egy gradiens módszerrel történő tanítási algoritmus ugyanazokat a változtatásokat hajtaná végre ezeken az egységeken. A teljes neurális hálóknál elterjedt módszer a normalizált inicializálás

$$w_{i,j} \sim E\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}} \right),$$

azaz egyenletes eloszlásból generáljuk a súlyokat, ahol m a neurális háló input változóinak, n a kimeneti változóinak a száma.

A neurális hálók hajlamosak a túltanulásra, amely során az empirikus rizikó az adathalmazon csökken, de az általánosítási hiba egy pont után növekedni kezd. A túltanulás monitorozható egy validációs adathalmaz fenntartásával, amelynek kiértékelését a tanítás során minden epoch után elvégezzük.

4. Opciók árazása neurális hálókkal

Ebben a fejezetben a különböző derivatív termékek árának neurális hálókkal való közelítésének eredményeit mutatom be. Piaci standardként, az opciókat tipikusan implikált volatilitásként jegyzik az opciós tőzsdéken, így a Black-Scholes formulánál a $\{S_0, K, r, \sigma, T-t\} \to P$ helyett, a $\{S_0, K, r, T-t, P\} \to \sigma$ iránynak is van gyakorlati jelentősége. Call és put opciónál, ezért az árazás mellett az implikált volatilitás meghatározására is tanítok neurális hálókat.

A 1 és 2 differenciálegyenletek a homogenitás miatt átskálázhatóak a K változó szerint, ezáltal a kötési árfolyam dimenzió elhagyható. Így az általánosság megszorítása nélkül elegendő az alaptermék árfolyamának és a kötési árfolyamnak a hányadosát legenerálni a szintetikus adathalmazokhoz. Ezt az S_0/K hányadost gyakran *moneyness*-ként nevezik. A K-val történő átskálázás továbbá azt is eredményezi, hogy az adathalmazban található változók nagyságrendje közel azonos lesz, amely a hálók könnyebb tanítása miatt előnyös tulajdonság.

Az neurális hálókat a Pytorch architektúra használatával implementáltam. [15] A hálók struktúrájához a mélység esetén

$$d \in \{16, 24, 32, 48, 64, 96, 128\},$$

szélességnél

$$k \in \{2, 4, 8, 16\},$$

a rejtett rétegek aktivációs függvényeinél

$$g(x) \in \{\tanh, \text{leaky-ReLU}, \text{GELU}\}$$

és a tanulási rátánál pedig az

$$\alpha \in \{0.001, 0.005, 0.01, 0.05\}$$

halmazokat használom. A háló kimeneti rétege minden modellnél egy neuron, amely az

opció értéke. Az ár pozitívitása, valamint folytonosan differenciálhatósága miatt aktivációs függvénynek a *SoftPlus* függvényt választom ezen a rétegen.

A hiperparaméter-optimalizálás során a paramétertérből egyenletes eloszlás szerint mintavételezek 20 hálóstruktúrát, amelyeket 10 epoch-on keresztül tanítok. A legjobb 10 validációs $L(X)$-el rendelkező struktúrát újratanítom 20 epoch-on keresztül, és ugyanígy kiválasztom az 5 legjobbat. A választott háló a legjobban teljesítő lesz ezek közül, 100 epoch-on történő tanítás után.

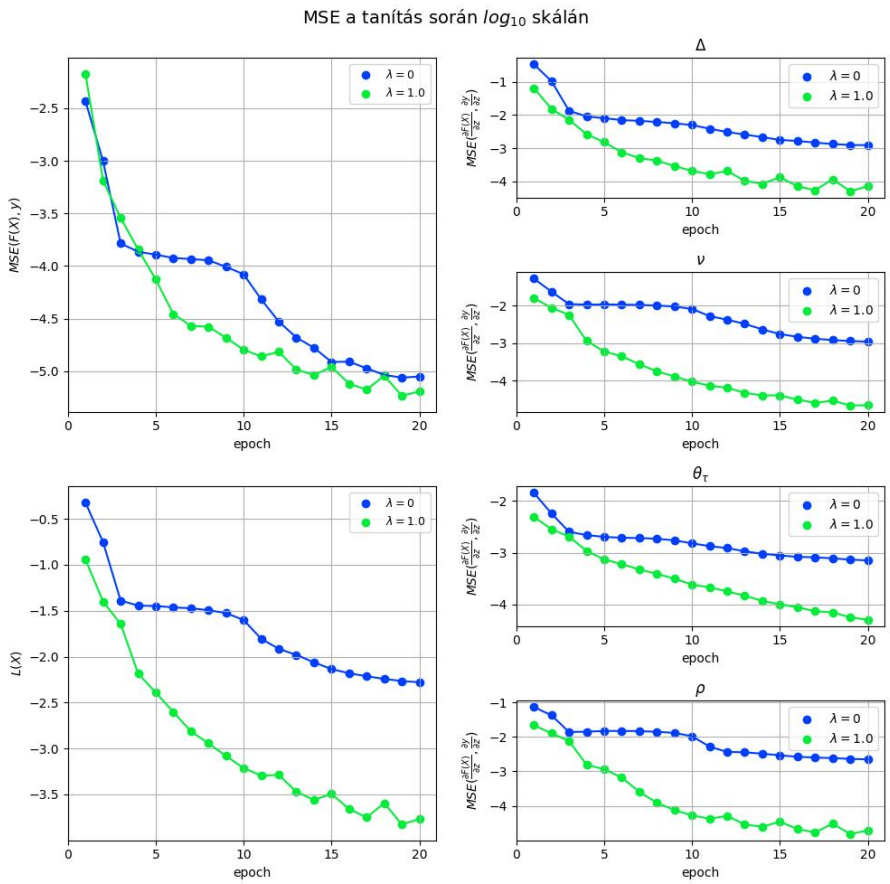
Tanításnál az adathalmazból minden epoch során véletlenül mintavételezek 10^5 adatpontot és ezeken $b = 2^8$ batch méretet használok. Tanító algoritmusnak a hálók az Adam algoritmust használják. Veszteségfüggvénynek az MSE-t választom, amelyhez hozzáadok a parciális deriváltak segítségével egy regularizációs tagot

$$L(X^j) = MSE(F(X^j), \underline{y}) + \lambda \sum_{z \in Z} \frac{1}{b} \sum_{i=1}^{b} \left(\frac{\partial F(\underline{X_i^j})}{\partial z} - \frac{\partial f(\underline{X_i^j})}{\partial z} \right)^2$$

ahol X^j a j-edik batch b batch mérettel, F a háló, f a derivatíva aminek árazására a hálót tanítom és Z az f paramétereinek egy halmaza. A λ paraméterrel a parciális deriváltak közelítésének fontossága állítható.[16]

Az automatikus differenciálás segítségével a $\frac{\partial F}{\partial z}(\underline{X_i^j})$ gyorsan számolható a tanítási lépésekben, a $\frac{\partial f}{\partial z}(\underline{X_i^j})$ - melyek legtöbbször görög betűk - értékei pedig az adatgenerálás során az árral együtt numerikusan vagy ha van analitikus képlet, akkor annak segítségével számolom és hozzáadom az adathalmazhoz. Így az adathalmaz $(X, \underline{y}, Z)$ hármassal adható meg, amelyeket véletlen sorba rendezés után b nagyságú batch-ekre felosztva használok a tanításhoz. Mivel a hálók inputként a $\tau = T - t$ értéket kapják meg, a piaci konvencióval szemben a θ görög betű helyett a θ_τ paricális deriváltat használom, amely a t helyett a τ szerinti parciális deriváltat jelöli.

Az 10 ábrán az MSE és az $L(X)$ hibák vannak ábrázolva az epochok számának függvényében, európai call opcióra esetén. Eleinte a hálók között az árazási hibát tekintve nincs számottevő eltérés viszont, ha a görög betűk hibáit is figyelembe vesszük, a regularizációval tanított háló később nagyságrendekkel jobban teljesít.



10. ábra. Regularizáció hatása a tanítás során európai call opció esetén.

A regularizáció használata enyhe torzítást okoz az árak predikcióiban, amelynek oka, hogy a tanításnál a függvény alakjának hibája jobban befolyásolja a paraméterek gradiensét, mint az árak eltérése.

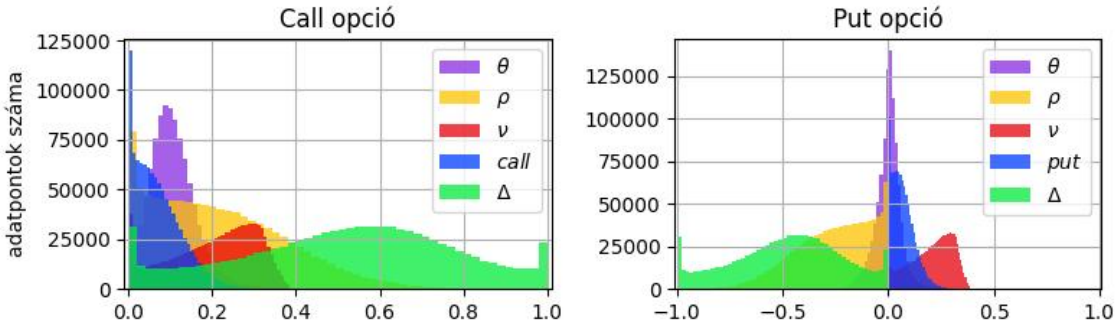
A hálóknál a továbbiakban minden esetben használom a regularizációt és a hiperparaméterek optimalizálásához a paramétertért kibővítem a $\lambda \in \{0.01, 0.1, 0.25, 0.5, 1.0, 2.0\}$ halmazzal.

4.1. Black-Scholes-Merton modell

4.1.1. Európai call és put opciók

Call és put opcióknál az adathalmaz X része 4 változóból áll: $(S_0/K, \tau, \sigma, r)$. Ahhoz, hogy az adathalmazban többségében olyan pontok legyenek, ahol az opció ATM környékén van, az S_0/K-t egyenletes eloszlás helyett az $S_0/K \sim \mathcal{N}(e^{-r\tau}, 0.01)$ normális eloszlás segítségével

generálom. Mivel a normális eloszlás tetszőleges értéket felvehet, a 0.1-nél kisebb vagy 1.9-nél nagyobb értékeket levágom. Az 11 ábra az adathalmazban szereplő ár és a görög betűk hisztogramjait mutatja put és call opció esetén.



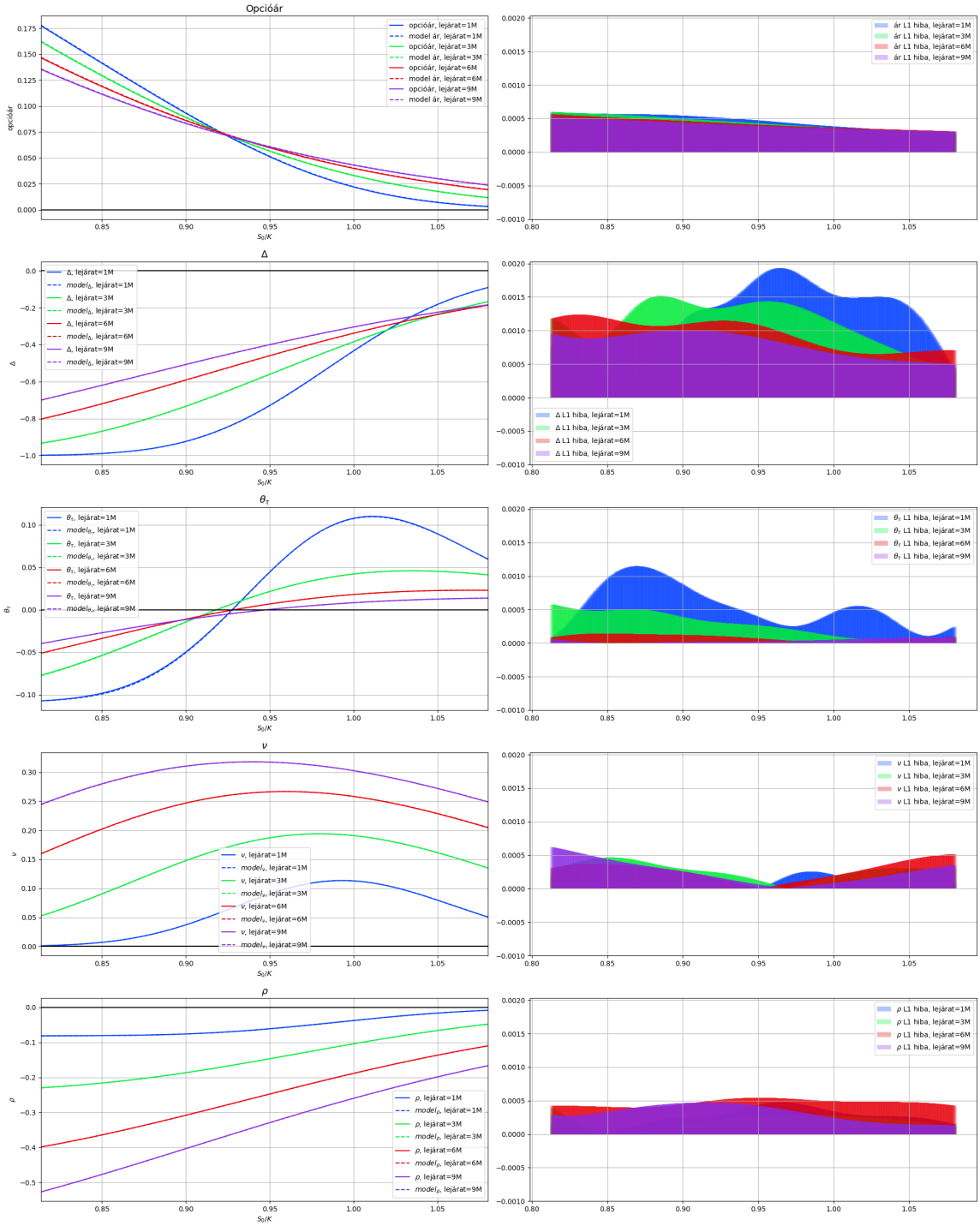
11. ábra. Az ár és a görög betük hisztogramja a tanító adathalmazokban

A regularizációhoz a görög betűk közül az összes elsőrendűt felhasználom az adathalmaz Z részében. Az analitikus formuláknál az adathalmaz generálása a vektorizálás miatt nem igényel hosszú futási időt. A görög betűk kiszámításával együtt is legfeljebb pár másodperc alatt elkészíthető akár 10^7 nagyságrendben.

	tanuló adathalmaz	**teszt adathalmaz**
S_0/K	$\mathcal{N}(e^{-r\tau}, 0.01)$	$\mathcal{N}(e^{-r\tau}, 0.0025)$
σ	$\mathcal{U}(0.1, 0.35)$	$\mathcal{U}(0.15, 0.25)$
r	$\mathcal{U}(0.01, 0.2)$	$\mathcal{U}(0.05, 0.15)$
τ	$\mathcal{U}(7/365, 1.0)$	$\mathcal{U}(14/365, 300/365)$

5. táblázat. Változók eloszlása az adatgenerálás során, put és call opciók esetén.

Az 12 ábrán a hiperparaméter-optimalizálás után választott háló predikciói vannak összevetve a tényleges értékekkel put opció esetén. Az opcióárakban kis mértékű torzítás figyelhető meg a regularizáció miatt, a görög betűk jó illeszkedést mutatnak.



12. ábra. Háló által prediktált értékek összehasonlítása a tényleges értékekkel, különböző lejáratokra. A háló egy belső pontban van kiértékelve, amely a tanulóadathalmaz változónkénti átlaga.

Az 6 táblázat a tanító és teszt adathalmazokon számolt MSE értékeket foglalja össze call és put opciók esetén az árra, valamint a görög betűkre.

		call		put	
		64 | 8 | tanh | $\lambda = 0.5$		**128 | 8 | tanh |** $\lambda = 1.0$	
		tanító adathalmaz	**teszt adathalmaz**	**tanító adathalmaz**	**teszt adathalmaz**
ár	MSE	1.6674e-07	1.2237e-07	8.9815e-07	6.4966e-07
Δ	MSE	1.1981e-05	1.0109e-05	9.1642e-06	5.8899e-06
θ_τ	MSE	4.8399e-06	8.6058e-07	9.6393e-06	1.8052e-06
ν	MSE	4.6260e-06	2.7756e-06	6.1357e-06	4.1319e-06
ρ	MSE	6.4467e-06	2.6898e-06	7.3239e-06	3.1663e-06

6. táblázat. Háló által számolt értékek hibája a tényleges értékekhez képest az MSE metrikában.

4.1.2. Implikált volatilitás

Az implikált volatilitás esetében nem szükséges új adathalmazt generálni, elegendő a call vagy a put opcióhoz is használt adatoknál az ár és a volatilitás változók szerepét megcserélni a tanításnál. A regularizációt itt csak az ár változó esetén lehet használni, amely az inverz függvény deriválási szabálya szerint

$$\frac{\partial c^{-1}}{\partial P}(P) = \frac{1}{\frac{\partial c}{\partial \sigma}(c^{-1}(P))} = \frac{1}{\nu_c(\sigma)} \tag{7}$$

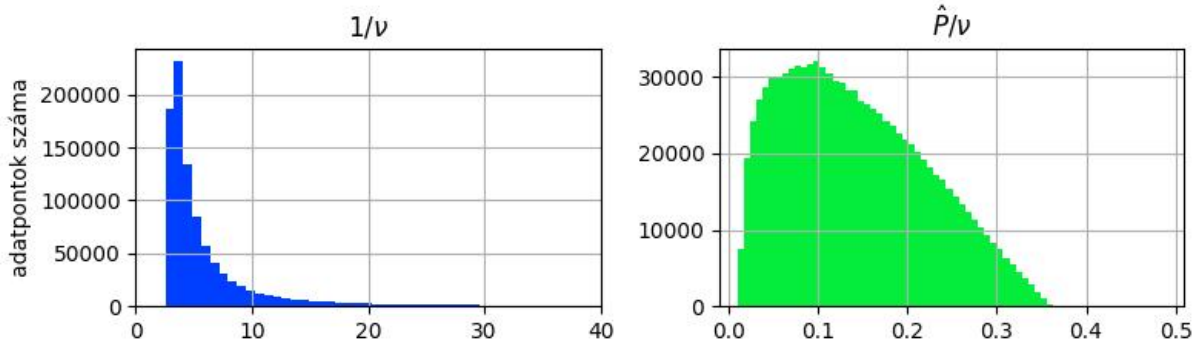
számolható. Így elég a vega értékek reciprokát venni az eredeti adathalmazban. Viszont nagyon OTM vagy ITM opciók esetén a vega értéke a 0 közelébe esik, amelyek a tanításnál a reciprok miatt instabillá tehetik a halót. Az opcióár megfelelő transzformálásával viszont a gradiensek nagyságrendje csökkenthető.[17] Legyen

$$\hat{P} = \log\left(P - \max(S_0/K - e^{-r\tau}, 0)\right),$$

vagyis az opció időértékének a logaritmusa. Azon adatpontokat melyeknél az időérték $< 10^5$, elhagyom az adathalmazból. Az 7 összefüggés alapján a regularizáció is használható marad, a láncszabály alapján

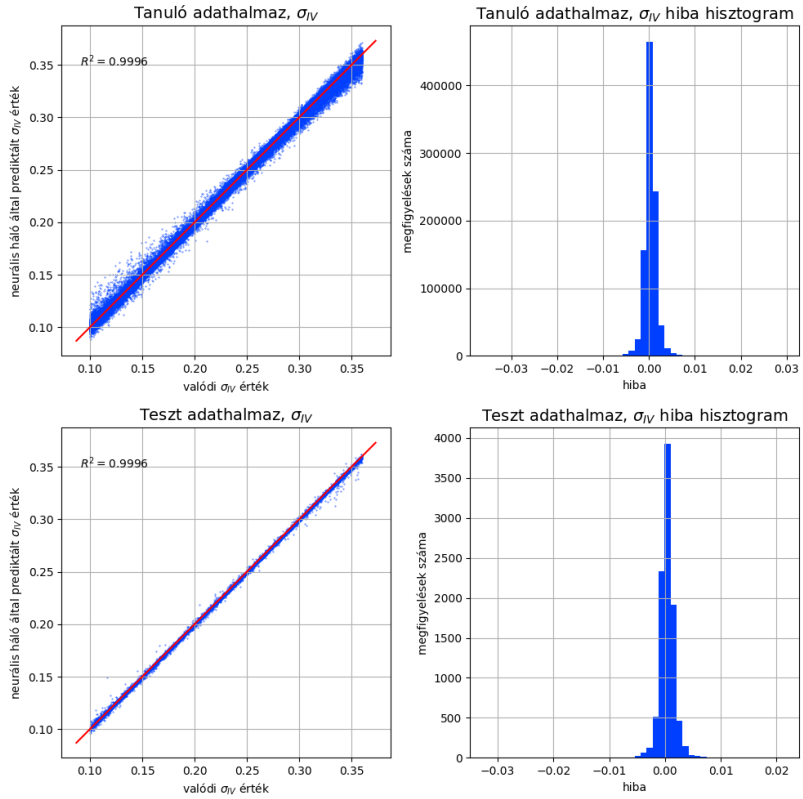
$$\frac{\partial c^{-1}}{\partial \hat{P}}(P) = \frac{\hat{P}}{\nu_c(\sigma)}.$$

Az 13 ábrán a P, valamint a $\hat{P}$ szerinti parciális deriváltak hisztogramjai szerepelnek a tanuló adathalmazból.



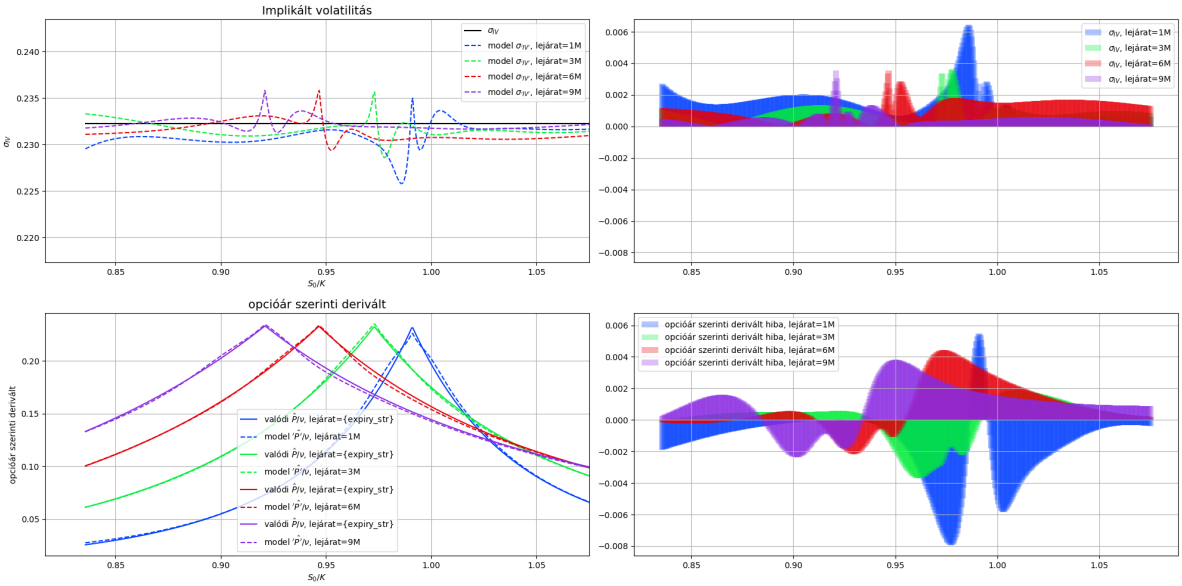
13. ábra. $\frac{\partial c^{-1}}{\partial P}$ és $\frac{\partial c^{-1}}{\partial \hat{P}}(P)$ hisztogramjainak összehasonlítása

Neurális hálókkal a transzformált árral történő tanítással jó illeszkedés érhető el, amelyet a 14 ábra szemléltet.



14. ábra. Neurális háló által prediktált értékek illeszkedése, valamint hibájának hisztrogramja a tanuló, illetve teszt adathalmazokon.

Az 15 ábrán az implikált volatilitás szerepel különböző lejáratokra, az S_0/K függvényében. Az adathalmaz szélein, ahol az opció nagyon ITM vagy OTM, a transzformálás után a háló közel ugyanakkora hibával prediktál, mint a sűrűbb belső pontokban. Legnagyobb eltérés a belső érték korrekció nem differenciálható pontja környékén figyelhető meg.



15. ábra. Háló által prediktált implikált volatilitás összehasonlítása a tényleges értékekkel, különböző lejáratokra. A háló a lejáraton kívül, egy belső pontban van kiértékelve, amely a tanulóadathalmaz változónkénti átlaga.

Az iteratív numerikus módszerekkel szemben, az implikált volatilitás kiszámítása a neurális hálóval rendkívül hatékony, a gyors kiértékelés miatt. A 4.2 fejezetben a Heston modell kalibrálásánál, ezért az ebben a fejezetben bemutatott hálót használom az implikált volatilitás számítására.

4.1.3. Barrier opciók

Az adathalmaz X része barrier opció esetén egy változóval bővül a B/K-val, így az adathalmaz az alábbi változókból áll: $(S_0/K, \tau, \sigma, r, B/K)$. A tanító adathalmaz 10^6 adatpontból áll, amelyek 10^5 különböző $\sigma, r, B/K$ paraméter hármasokból számolt árazó rácsokból származnak. Az árakat és görög betűket a 2.4 alfejezetben bemutatott Crank-Nicolson módszer segítségével generálom. Az adathalmazok generálásának futásideje 100×100 dimenziós árazó rács esetén ≈ 10 perc, a görög betűk kiszámolásával együtt.

	up barrier		down barrier	
	tanuló adathalmaz	teszt adathalmaz	tanuló adathalmaz	teszt adathalmaz
S_0/K	$\mathcal{U}(0.6 \cdot B/K, B/K - 0.001)$	$\mathcal{U}(0.7 \cdot B/K, B/K - 0.001)$	$\mathcal{U}(B/K + 0.001, B/K \cdot 1.4)$	$\mathcal{U}(B/K + 0.001, B/K \cdot 1.3)$
σ	$\mathcal{U}(0.1, 0.35)$	$\mathcal{U}(0.15, 0.25)$	$\mathcal{U}(0.1, 0.35)$	$\mathcal{U}(0.15, 0.25)$
r	$\mathcal{U}(0.01, 0.2)$	$\mathcal{U}(0.05, 0.15)$	$\mathcal{U}(0.01, 0.2)$	$\mathcal{U}(0.05, 0.15)$
τ	$\mathcal{U}(60/365, 1.0)$	$\mathcal{U}(75/365, 300/365)$	$\mathcal{U}(60/365, 1.0)$	$\mathcal{U}(75/365, 300/365)$
B/K	$\mathcal{U}(1.1, 1.8)$	$\mathcal{U}(1.2, 1.7)$	$\mathcal{U}(1/1.8, 1/1.1)$	$\mathcal{U}(1/1.7, 1/1.2)$

7. táblázat. Változók eloszlása az adatgenerálás során.

A lejáratig hátralévő idő csökkentésével a B/K alaptermék árfolyam környezetében az opcióár nagyon meredeken tart a B/K pontban felvett értékhez. A mély neurális hálózatok a paramétertér valamely tengelyén magas meredekséggel rendelkező tartományaira nehezen, vagy nagyobb hibával képesek rátanulni. [17] Ezért az adathalmazokban a τ változó a call és put opciók képest szűkebb intervallumokból van generálva. Továbbá az opció árához képest a a Δ akár több nagyságrenddel is nagyobb értékeket vehet fel. Emiatt az árban megjelenő torzítás csökkentése miatt a λ regularizáció paramétert is kisebb értékekből választom a hiperparaméterek optimalizálása során.

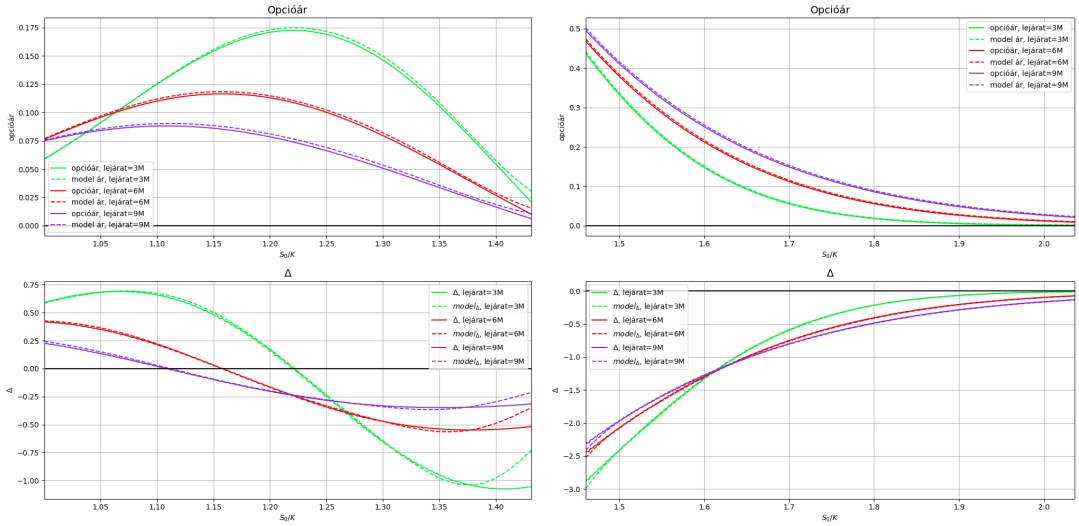
Call és put opció árainak meghatározása mellett a 8 különböző típusú barrier call és put opció közül elegendő páronként egyet meghatározni. Ugyanis replikálás segítségével a többi termék ára is előállítható. Az 8 táblázat a tanító és teszt adathalmazokon számolt MSE értékeket foglalja össze call és put opciók esetén az árra, valamint a görög betűkre.

		up-and-out call 24 | 4 | GeLU | $\lambda = 0.5$		up-and-in put 64 | 4 | GeLU | $\lambda = 0.5$		down-and-out put 24 | 4 | tanh | $\lambda = 0.5$		down-and-in call 32 | 4 | tanh | $\lambda = 0.25$	
		tanító adathalmaz	teszt adathalmaz	tanító adathalmaz	teszt adathalmaz	tanító adathalmaz	teszt adathalmaz	tanító adathalmaz	teszt adathalmaz
ár	MSE	6.5866e-06	1.4766e-06	1.9968e-05	2.0811e-05	1.7114e-04	1.6108e-04	1.1134e-05	2.1971e-05
Δ	MSE	2.5564e-04	2.7720e-03	3.7694e-03	3.9348e-03	6.0353e-03	1.1902e-03	1.8077e-04	7.2573e-03
θ_τ	MSE	1.0933e-04	4.1416e-05	1.4352e-04	1.2186e-04	4.4874e-04	2.2252e-04	7.0181e-05	3.3254e-04
ν	MSE	1.5533e-03	1.8574e-04	8.2271e-04	5.1513e-04	1.2319e-03	6.1958e-04	1.8780e-04	1.4602e-03
ρ	MSE	8.9164e-05	5.6498e-05	3.5465e-04	2.1383e-04	3.5572e-04	1.6184e-04	7.4634e-05	1.0053e-03

8. táblázat. Háló által számolt értékek hibája a tényleges értékekhez képest az MSE metrikában barrier opciókra.

Az 16 ábrán is megfigyelhető, hogy a legnagyobb eltérések az árban és a Δ-ban is egyaránt

a B/K pont környékén találhatóak a háló predikcióinál.



16. ábra. Háló által prediktált értékek összehasonlítása a tényleges értékekkel, különböző lejáratokra $c^{UO}_{1,B/K}$ és $c^{DI}_{1,B/K}$ opciók esetén.

4.2. Modellkalibráció neurális hálókkal

4.2.1. Heston modell call és barrier opciók

Az adathalmaz X része call opció esetén 8 változóból áll: $(S_0/K, \tau, V_0, r, \rho, \kappa, \theta, \sigma)$, barrier opció esetén pedig a B/K is adathalmaz része. Az adatgenerálás a Heston modell esetén lényegesen lassabb, mint a BSM modellben a barrier opcióknál. Tanító adathalmazhoz egy modell paraméterezésből 100 különböző $(S_0/K, \tau, V_0)$ pontra határozom meg az árat. Így a Feller-feltétel teljesülése mellett összesen 10^4 különböző paraméterezésből 10^6 adatpont szerepel az adathalmazban.

	call		up-and-out barrier call	
	tanuló adathalmaz	teszt adathalmaz	tanuló adathalmaz	teszt adathalmaz
S_0/K	$\mathcal{N}(e^{-r\tau}, 0.01)$	$\mathcal{N}(e^{-r\tau}, 0.0025)$	$\mathcal{U}(0.6 \cdot B/K, B/K - 0.001)$	$\mathcal{U}(0.7 \cdot B/K, B/K - 0.001)$
V_0	$\mathcal{U}(\theta \cdot 0.75, \theta \cdot 1.25)$	$\mathcal{U}(\theta \cdot 0.75, \theta \cdot 1.25)$	$\mathcal{U}(\theta \cdot 0.75, \theta \cdot 1.25)$	$\mathcal{U}(\theta \cdot 0.75, \theta \cdot 1.25)$
r	$\mathcal{U}(0.01, 0.1)$	$\mathcal{U}(0.01, 0.08)$	$\mathcal{U}(0.01, 0.1)$	$\mathcal{U}(0.01, 0.08)$
τ	$\mathcal{U}(14/365, 1.0)$	$\mathcal{U}(30/365, 1.0)$	$\mathcal{U}(120/365, 1.0)$	$\mathcal{U}(150/365, 1.0)$
ρ	$\mathcal{U}(-0.5, 0.1)$	$\mathcal{U}(-0.4, 0.1)$	$\mathcal{U}(-0.5, 0.1)$	$\mathcal{U}(-0.4, 0.1)$
κ	$\mathcal{U}(1.0, 4.0)$	$\mathcal{U}(1.5, 3.5)$	$\mathcal{U}(1.0, 4.0)$	$\mathcal{U}(1.5, 3.5)$
θ	$\mathcal{U}(0.1^2, 0.25^2)$	$\mathcal{U}(0.125^2, 0.225^2)$	$\mathcal{U}(0.1^2, 0.25^2)$	$\mathcal{U}(0.125^2, 0.225^2)$
σ	$\mathcal{U}(0.1, 0.6)$	$\mathcal{U}(0.2, 0.5)$	$\mathcal{U}(0.1, 0.6)$	$\mathcal{U}(0.2, 0.5)$
B/K	-	-	$\mathcal{U}(1.1, 1.8)$	$\mathcal{U}(1.2, 1.7)$

9. táblázat. Változók eloszlása a Heston modellból származó adatok generálása során.

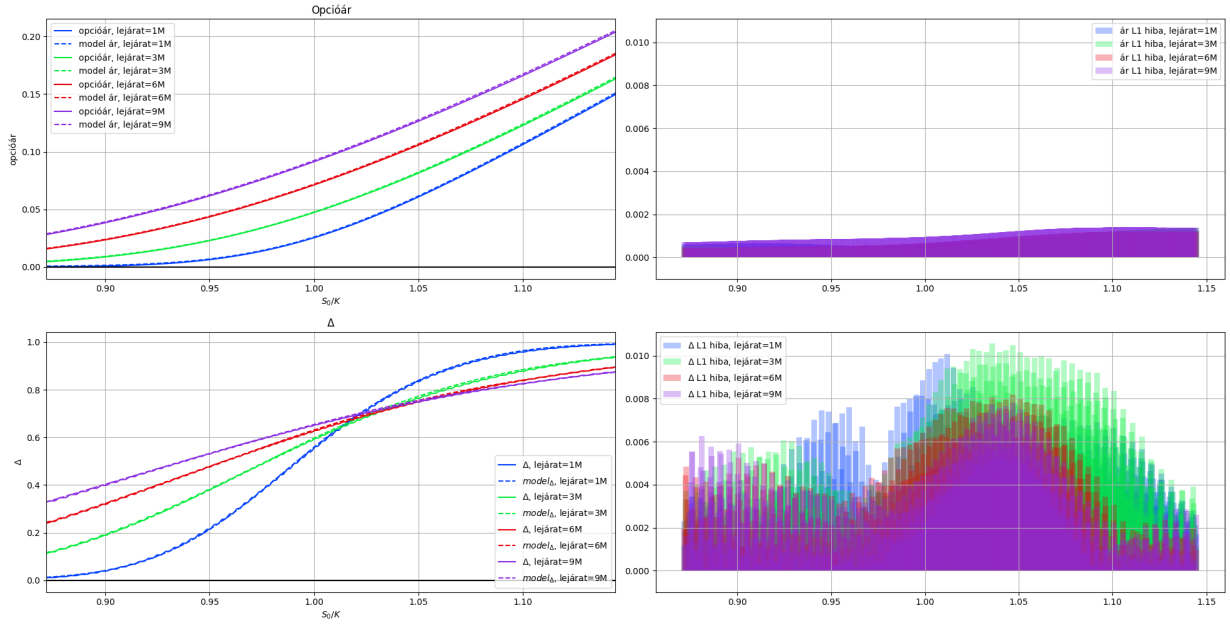
Az adathalmaz Z részében mindegyik változó szerinti parciális derivált szerepel a B/K változón kívül, kompenzálva bizonyos árazási paraméterek alacsonyabb szórását. Emiatt az adatgenerálás futásideje a parciális deriváltak meghatározásával együtt 4 processzormagon ≈ 2 órára növekszik, $50 \times 100 \times 100$ felosztású rács mellett.

Az 11 táblázat a tanító és teszt adathalmazokon számolt MSE értékeket foglalja össze call és up-and-out barrier call opció esetén az árra, valamint a görög betűkre. Az eltérések nagyságrendje a több input változó, valamint a paraméterek kisebb szórása miatt nagyobb a BSM modellen tanított neurális hálókéhoz képest.

		Heston call		Heston up-and-out call	
		64 | 8 | tanh | $\lambda = 1.0$		128 | 2 | GeLU | $\lambda = 0.1$	
		tanító adathalmaz	teszt adathalmaz	tanító adathalmaz	teszt adathalmaz
ár	MSE	5.4437e-05	4.8597e-05	1.1150e-03	1.2622e-03
Δ	MSE	7.4685e-04	7.8749e-04	3.4645e-02	2.6031e-02
θ_τ	MSE	1.5911e-04	1.5379e-04	1.8282e-03	1.6843e-03
ν	MSE	1.5990e-03	1.4738e-03	4.4343e-02	4.4459e-02
ρ	MSE	2.3128e-04	2.3322e-04	6.0044e-03	6.1415e-03
	$\frac{\partial f}{\partial \text{corr}}$	1.2530e-06	1.1452e-06	9.5922e-05	1.1442e-04
	$\frac{\partial f}{\partial \kappa}$	3.9682e-07	4.0224e-07	6.7190e-06	9.5121e-06
	$\frac{\partial f}{\partial \text{theta}}$	1.7193e-03	1.5500e-03	3.5093e-02	4.0878e-02
	$\frac{\partial f}{\partial \sigma}$	4.5582e-06	4.2644e-06	2.4680e-04	3.0416e-04

10. táblázat. Háló által számolt értékek hibája a tényleges értékekhez képest az MSE metrika szerint.

Az 17 ábrán a vizsgált paramétertérből a legjobb hiperparaméterekkel rendelkező neurális háló predikciói vannak összevetve a tényleges értékekkel, call opció esetén. Az opcióárakban ebben az esetben is kisebb mértékű torzítás figyelhető meg a regularizáció miatt.



17. ábra. Háló által prediktált ár és Δ értékek összehasonlítása a tényleges értékekkel, különböző lejáratokra. A neurális háló egy belső pontban van kiértékelve, amely a tanuló-adathalmaz változónkénti átlaga.

4.2.2. Kalibráció

A kalibrálás célja olyan modell paraméterek meghatározása, amelyek mellett a piacon megfigyelt és a modellből számolt implikált volatilitás értékek közötti távolság a lehető legkisebb legyen valamilyen metrika szerint. Gyakran a likviditás vagy egyéb súlyozás szerint az egyes termékek különböző súllyal vesznek részt a kalibrálásban, ezáltal a likvidebb lejáratokra és kötési árfolyamokra a kalibrált modell pontosabban áraz.

A neurális hálók tanításához hasonlóan ez a feladat is nem konvex optimalizáláshoz vezet. A kalibrálás során a távolságmetrikának az MSE-t választom, és minden terméket a piacon azonos súllyal veszek figyelembe. Így a minimalizálandó célfüggvény

$$\sum_{T,K} \left(\sigma_{IV}(K,T) - \hat{\sigma}(K,T)\right)^2,$$

vagy

$$\sum_{T,K} \left(c(K,T) - \hat{c}(K,T)\right)^2$$

alakban írható fel az implikált volatilitás invertáhatósága alapján.

A Heston modell esetében a kalibrálásban a $(V_0, \rho, \kappa, \theta, \sigma)$ paraméterek vesznek részt. Ezeket a paramétereket a neurális hálók segítségével, különböző szintetikusan generált piacokra kalibrálom. Háromféleképp is tesztelem a neuronhálókkal történő kalibrálást:

1. A piacon csak call opciókkal kereskednek, és a kalibrálást az implikált volatilitás szerint végzem el.

2. A piacon csak call opciókkal kereskednek, és a kalibrálást az ár szerint végzem el.

3. A piacon kereskednek call, valamint up-and-out barrier call opciókkal is. A kalibrálást a call opciók esetén az implikált volatilitás, a barrier opciók esetén az ár szerint végzem el.

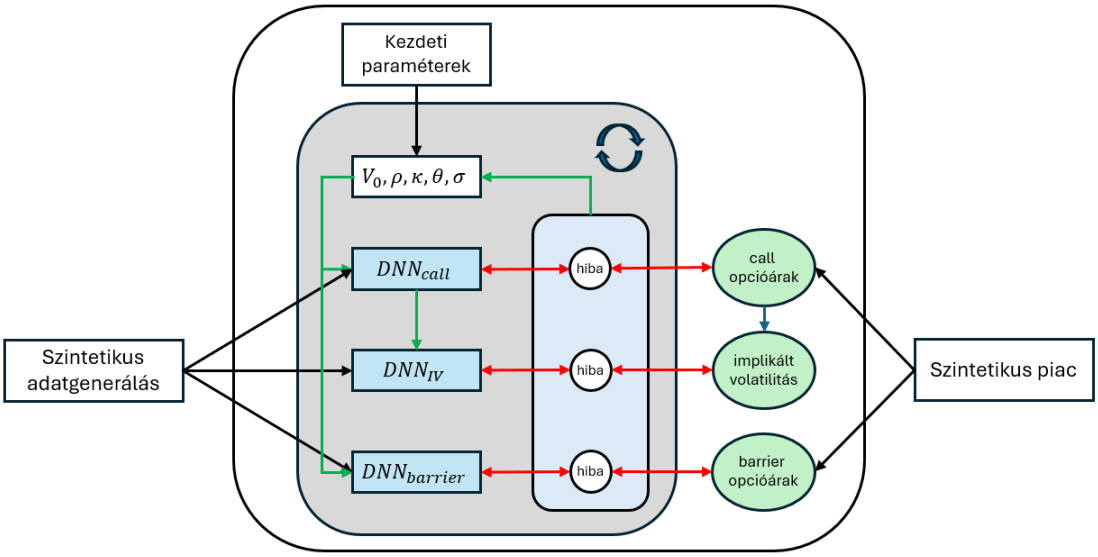
A barrier opciók figyelembevétele a kalibrációban pontosabb paraméterekhez vezet, mivel ezen termékek az útvonalfüggőség révén érzékenyebbek a volatilitás változására a trajektóriák mentén. Ezáltal a jövőbeni mosoly és ferdeség (*forward smile and skew*) dinamikáját a kalibrált modell pontosabban képes visszaadni.[2]

A szintetikus piaci adatokat szintén a Heston modell alapján generálom. A piacokon a kockázatmentes hozam $r = 0.03$ százalék, call opciókkal

$$T \in \{3\text{M}, 4\text{M}, 5\text{M}, 6\text{M}, 7\text{M}, 9\text{M}, 12\text{M}\}$$

lejáratokon lehet kereskedni az $S_0/K = e^{-rT} \pm 0.15$ között egyenletesen 7 pontban felosztott kötési árfolyamokon. A barrier opciókból kétféle lejárattal lehet kereskedni, egy 6 hónap lejáratúval és egy 10 hónap lejarátúval az $S_0/K \in \{1.05, 1.15\}$ kötési árfolyamokon.

Az árakat a 2.4.4 alfejezetben bemutatott CS-ADI módszerrel számolom, a kalibrációban részt vevő paramétereket a neurális hálók tanuló adathalmazai szerinti eloszlás szerint mintavételezem. Összesen 100 különböző szintetikus piacra kalibrálom a paramétereket a neurális hálók segítségével. A 18 ábra az implementált kalibrációs eljárás folyamatábráját szemlélteti.

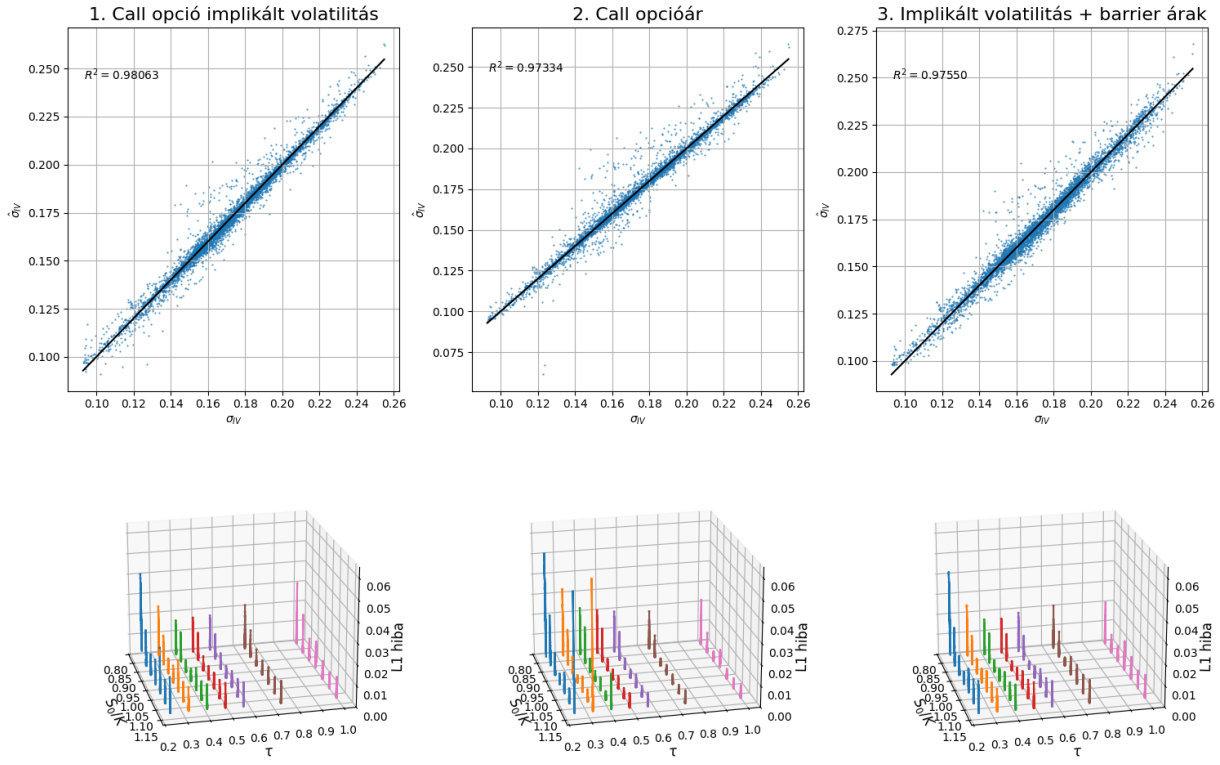


18. ábra. Neurális hálókkal történő kalibrálás folyamatábrája.

A kezdeti paramétereket a valódi paraméterekkel megegyező eloszlások szerint határozom meg. Optimalizáló eljárásnak pedig *Jim Gatheral* és *Peter Pommergård Lind* cikke alapján a Nelder-Mead algoritmust használom, amely az iterációs lépéseiben nem használ gradienseket. A Nelder-Mead algoritmus bár lassabb, de nem akad el egy nyeregpontban vagy lokális minimum helyen, mint általában a gradiensalapú BFGS vagy L-BFGS-B algortimusok. [18]

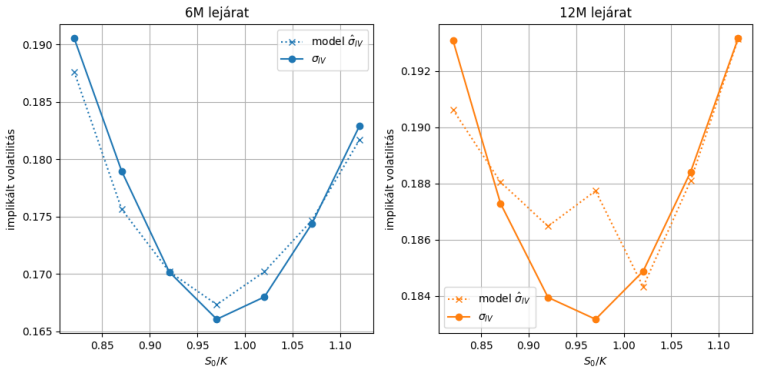
Az 19 ábra a különböző célfüggvények szerinti kalibrálás eredményeit foglalja össze. A

legnagyobb eltérések a legrövidebb lejáratnál figyelhetőek meg, az OTM és az ITM kötési árfolyamokra. Ennek az oka a neurális háló regularizációból eredő torzítása, amely miatt a háló nem tud tetszőlegesen alacsony árat prediktálni. Így ez a kis eltérés, az alacsonyabb időérték és az $1/\nu$ parciális derivált miatt az implikált volatilitásban potenciálisan nagyobb eltérésekhez vezet.



19. ábra. Implikált volatilitások illeszkedése és hibája a kalibrálás után.

Az egy éves lejárattal rendelkező opciók a tanító adathalmaz szélénél helyezkedkednek el, így rajtuk a neurális háló extrapoláló képessége is tesztelhető. A neurális hálóknak a tanítási inputtéren kívül általában gyengén extrapolálnak, amely ebben az esetben is fenn áll. Ennél a lejáratnál a háló szinte egyik piac esetén sem képes még az implikált volatilitásfelület alakját se reprodukálni. Az 20 ábra a hat hónapos és az egy éves lejáratra vonatkozó valódi és a modell által számolt implikált volatilitásokat hasonlítja össze az egyik szintetikus piac esetén.



20. ábra. Neurális háló extrapoláló képessége. A 6M lejárat a hálók tanító adathalmazának a belsejében, míg a 12M lejárat a határon található

A piacon megfigyelt implikált volatilitás értékeket a neurális háló a legtöbb vizsgált piac esetében alacsony hibával képes reprodulálni, de ez nem igaz az piacot generáló eredeti paraméterekre. A ρ, κ és σ paraméterek esetében akár többszörös eltérés is előfordulhat.

	V_0	ρ	κ	θ	σ
átlagos $L1$ hiba	0.0030	0.1458	1.1402	0.0043	0.2822

11. táblázat. Neurális hálók által kalibrált paraméterek hibája a tényleges generáló paraméterekhez képest.

A kalibrálás futásideje átlagosan az 1. esetben 3.2 mp, a 2. esetben 3.41 mp, míg a barrier árakat is használó 3. esetben 4.37 mp. Így a neurális hálókkal történő kalibrálás során a barrier opciók használata nem okoz lényegi futásidő növekedést.

5. Összefoglalás

A dolgozathoz használt Python kódbázist struktúráltan és objektumorientált szemléletettel sikerült implementálni. Ahol a pénzügyi termékek árazása, a szintetikus adatok generálása, valamint a neurális hálók tanítása és kiértékelése is ugyanaz a sztenderdizált architektúra alapján épül fel. Amelynek futtatása rugalmasan elvégezhető a konfigurációs fájlok segítségével. A kódbázishoz így könnyedén hozzáadhatóak akár más termékek, és általánosítható egyéb modellekre egyaránt.

A neurális hálók tanítása során olyan regularizációt alkalmazok, amely a parciális deriváltak hibáját is figyelembe veszi minden lépésben. Ezáltal a neuronhálók az opciók árai mellett a görög betűk értékeit is képesek pontosan és hatékonyan visszaadni az automatikus differenciálás felhasználásával. A regularizációval emellett csökkenthető a túltanulás veszélye, valamint lehetővé tesz egyszerűbb háló struktúrák használatát, amely lényegesen felgyorsítja a neurális hálók tanítását.

A Black-Scholes-Merton modellben az európai call, put és barrier opciók árát és a görög betűk értékeit a hálók rendkívül jó pontossággal képesek visszaadni. Az adathalmaz transzformálása után hasonló pontosságot sikerült elérni az implikált volatilitás esetében is.

A neurális hálók a Heston modellben is rá tudtak tanulni az árra és a görög betűkre, call és barrier opciók esetén egyaránt. A számítási kapacitás limitációi miatt az adathalmazok egyes változókat kisebb szórással tartalmaznak, ezáltal a neuronhálók hibája is nagyobb a Black-Scholes-Merton modellhez képest.

A dolgozat egyik célkitűzése a Heston modell paramétereinek neurális hálókkal történő kalibrálása volt. A kalibrációs eljárás sikeresnek mondható abból a szempontból, hogy a hálók képesek visszaadni a piacon megfigyelt implikált volatilitásfelületet, akár a barrier opcióárak figyelembevételével is. A szintetikus piacokat generáló valódi paramétereket viszont a neurális hálók csak megközelítőleg képesek reprodukálni.

5.1. Továbbfejlesztési lehetőségek

A Heston modellnél a véges differenciákkal történő adatgenerálás Pythonban történő implementálása nagyon lassú futásidőket eredményez. A kódbázis ezen részének refaktorálása egy

hatékonyabb hardverközeli programozási nyelvre - mint a C++ - akár többszörös teljesítmény növekedést is eredményezhet.

A véges differenciák módszerén alapuló algoritmusok elméletben ugyanúgy implementálhatóak egy automatikus differenciálást alkalmazó architekturán. A nehezséget az interpoláció, és az együtthatómátrixok invertálása vagy LU-felbontása jelenti, valamint az, hogy a ritka mátrixokat egyáltalán nem, vagy csak nehezen lehet kezelni ezekben az architektúrákban. A jelenlegi numerikus közelítés helyett, a backpropagation használatával elegendő lenne az árazási rácsot csak egyszer kiszámolni, amely során megkaphatnánk az ár mellett egyben az összes inputparaméter szerinti parciális deriváltat is.

Az árazási felületet másik megközelítésben is meg lehet tanítani a neurális hálókkal. A Fourier hálózati operátoroknál a háló az Euklideszi terek között értelmezett függvények helyett függvényterek közti leképezéseket képes megtanulni. [19] Ezáltal felosztás-függetlenül lehet sztochasztikus differenciálegyeneletek teljes családját egyetlen neurális hálóval közelíteni, hiszen a háló a differenciálegyenlet struktúrájára tanul rá nem pedig az egyedi adatpontokra.

Alulírott Halász Kristóf nyiltakozom, hogy szakdolgozatom elkészítése során az alább felsorolt feladatok elvégzésére a megadott MI alapú eszközöket alkalmaztam:

Feladat	**Felhasznált eszköz**	**Felhasználás helye**	**Megjegyzés**
Python kódbázis elkészítése	Github Copilot	Teljes kódbázis	Automatikus kiegészítés funkció Python kód írása közben

A felsoroltakon túl más MI alapú eszközt nem használtam.

Hivatkozások

[1] Aitor Muguruza Blanka Horvath és Mehdi Tomas. „Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models”. *Quantitative Finance* 21.1 (2021), 11–27. old. DOI: 10.1080/14697688.2020.1817974. eprint: https://doi.org/10.1080/14697688.2020.1817974. URL: https://doi.org/10.1080/14697688.2020.1817974.

[2] Jim Gatheral. *The volatility surface: a practitioner's guide*. John Wiley & Sons, 2011.

[3] Bank for International Settlements. *OTC derivatives statistics at end-June 2024*. 2024. URL: https://www.bis.org/publ/otc_hy2411.htm.

[4] Fischer Black és Myron Scholes. „The Pricing of Options and Corporate Liabilities”. *Journal of Political Economy* 81.3 (1973), 637–654. old. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831029.

[5] Robert C. Merton. „Theory of Rational Option Pricing”. *The Bell Journal of Economics and Management Science* 4.1 (1973), 141–183. old. ISSN: 00058556, 23255323. URL: http://www.jstor.org/stable/3003143.

[6] J.C. Hull. *Options, Futures, and Other Derivatives, Global Edition*. Pearson Education, 2021. ISBN: 9781292410623.

[7] Márkus László. *Pénzügyi folyamatok matematikája II. (Előadásjegyzet)*. 2014.

[8] E. Reiner és M. Rubinstein. „Breaking Down the Barriers”. *Risk Magazine* 4.8 (1991), 28–35. old.

[9] Paul Wilmott. *Paul Wilmott on quantitative finance*. John Wiley & Sons, 2013. ISBN: 9781118836835.

[10] In'T Hout, S Foulon és tsai. „ADI finite difference schemes for option pricing in the Heston model with correlation.” *International Journal of Numerical Analysis & Modeling* 7.2 (2010).

[11] Ian Goodfellow, Yoshua Bengio és Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[12] Kurt Hornik, Maxwell Stinchcombe és Halbert White. „Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". *Neural networks* 3.5 (1990), 551–560. old.

[13] Yann N Dauphin és tsai. „Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". *Advances in neural information processing systems* 27 (2014).

[14] Diederik P Kingma és Jimmy Ba. „Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014). DOI: https://doi.org/10.48550/arXiv.1412.6980.

[15] A Paszke. „Pytorch: An imperative style, high-performance deep learning library". *arXiv preprint arXiv:1912.01703* (2019).

[16] Thomas O'Leary-Roseberry és tsai. „Derivative-informed neural operator: an efficient framework for high-dimensional parametric derivative learning". *Journal of Computational Physics* 496 (2024), 112555. old.

[17] Shuaiqiang Liu, Cornelis W. Oosterlee és Sander M. Bohte. „Pricing Options and Computing Implied Volatilities using Neural Networks". *Risks* 7.1 (2019. febr.), 16. old. ISSN: 2227-9091. DOI: 10.3390/risks7010016. URL: http://dx.doi.org/10.3390/risks7010016.

[18] Peter Pommergård Lind és Jim Gatheral. „NN de-Americanization: an efficient method to facilitate calibration of American-style options". *Quantitative Finance* 25.1 (2025. jan.), 1–16. old. DOI: 10.1080/14697688.2024.2432511. URL: https://ideas.repec.org/a/taf/quantf/v25y2025i1p1-16.html.

[19] Zongyi Li és tsai. „Fourier neural operator for parametric partial differential equations". *arXiv preprint arXiv:2010.08895* (2020).