

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1951

**RAZVOJ APLIKACIJE ZA PROCJENU RIZIKA U
INVESTICIJSKIM PORTFELJIMA UZ POMOĆ
MONTE CARLO SIMULACIJA**

Ivan Džanija

Zagreb, lipanj, 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

ZAVRŠNI ZADATAK br. 1951

Pristupnik: **Ivan Džanija (0036547433)**

Studij: Elektrotehnika i informacijska tehnologija i Računarstvo

Modul: Računarstvo

Mentorica: izv. prof. dr. sc. Mihaela Vranić

Zadatak: **Razvoj aplikacije za procjenu rizika u investicijskim portfeljima uz pomoć Monte Carlo simulacija**

Opis zadatka:

Monte Carlo simulacije su moćan alat koji investitorima pomaže u procjeni rizika i potencijalnih povrata na ulaganja u investicijskim portfeljima. Ova metoda omogućuje investitorima da bolje razumiju raspon mogućih ishoda i donose informiranije odluke o alokaciji imovine. Cilj ovog završnog rada je razviti aplikaciju koja implementira Monte Carlo simulacije za procjenu rizika i potencijalnih povrata u investicijskim portfeljima. Aplikacija treba koristiti javno dostupne podatke o kretanju cijena različitih vrijednosnih papira te omogućiti korisnicima analizu osjetljivosti portfelja na promjene tržišnih uvjeta. Ključne funkcionalnosti uključuju generiranje velikog broja simulacija kako bi se procijenio raspon mogućih ishoda, vizualizaciju rezultata u obliku distribucija i ključnih metrika rizika te intuitivno korisničko sučelje koje olakšava donošenje informiranih odluka o alokaciji imovine. U sklopu rada potrebno je detaljno dokumentirati razvijenu aplikaciju, prikazati reprezentativne slučajeve uporabe, te kritički analizirati prednosti i ograničenja razvijenog rješenja. Radu treba priložiti izvorni programski kod, korištene skupove podataka i upute za korištenje aplikacije.

Rok za predaju rada: 23. lipnja 2025.

Hvala na svemu puno. ovo je test i opet

Sadržaj

1. Uvod	3
2. Teorija portfelja i matematičke osnove	5
2.1. Investicije	5
2.2. Portfelj	5
2.3. Povrati	6
2.3.1. Aritmetički povrat	6
2.3.2. Logaritamski povrat	6
2.3.3. Očekivani povrat	6
2.4. Volatilnost	7
2.5. Geometrijsko Brownovo gibanje	7
2.6. Monte Carlo simulacije	8
2.7. Cholesky dekompozicija	9
2.8. PCA dekompozicija	9
3. Implementacija aplikacije	11
3.1. Podatci	11
3.2. Algoritmi i komponente	13
3.2.1. Parsiranje podataka	13
3.2.2. Model kriptovalute	14
3.2.3. Model portfelja	15
3.2.4. Matematički okvir	16
3.2.5. Monte Carlo simulacija	20
3.2.6. PCA dekompozicija	20
3.2.7. Vizualizacija rezultata	21

4. Rezultati i rasprava	23
4.1. Implementacijski rezultati	23
4.1.1. Funkcionalnost učitavanja i parsiranja podataka	23
4.1.2. Matematički okvir	23
4.1.3. Monte Carlo simulacija	24
5. Zaključak	25
Literatura	26
Sažetak	28
Abstract	29
A: The Code	30

1. Uvod

Modeliranje ponašanja portfelja je jedna od ključnih metoda pri odabiru investicijskog pristupa. Od odabira pojedinačnih dionica, raznih derivata, sigurnih obveznica ili investicijskih fondova sve do mogućnosti čuvanja novčanih rezervi.

U posljednjem desetljeću, kriptovalute su postale sveprisutna komponenta financijskih tržišta, karakterizirana visokom volatilnošću, nelinearnim ovisnostima i globalnom dostupnošću što kroz iznimno pouzdane izvore što kroz izrazito nepouzdane izvore. Upravljanje rizikom u takvom okruženju zahtijeva napredne alate za modeliranje budućih scenarija. Jedan od takvih alata u standarnim financijskim modelima je Monte Carlo simulacija.

Monte Carlo simulacija, kao statistička metoda temeljena na ponovljenom uzorkovanju slučajnih varijabli i često korištena metoda u modeliranju ostalih financijskih instrumenata, nameće se kao potencijalno uspješan pristup za analizu portfelja kriptovaluta.

U ovom radu fokusira se na implementaciji Monte Carlo metode za predviđanje vrijednosti portfelja s primjenom Cholesky dekompozicije kako bi se osigurala realistična obrada korelacija između kriptovaluta koje su još uvijek specijalna skupina investicija s visokom međusobnom korelacijom.

Osim Monte Carlo simulacija za predviđanje budućih cijena portfelja, implementira se i PCA (Principal Component Analysis) dekompozicija kao jedna od jednostavnih metoda za smanjenje volatilnosti portfelja upravo postignutu kroz smanjenje dimenzionalnosti podataka i identifikaciju glavnih komponenti koje objašnjavaju najveći dio varijance podataka.

Glavni izazov u modeliranju kriptovaluta leži u njihovoј inherentnoj nestabilnosti. Dok tradicionalne financijske instrumente karakteriziraju relativno predvidljivi obrasci, kriptovalute pokazuju ekstremne fluktuacije koje zahtijevaju precizno podešavanje parametara poput driftova i volatilnosti. Osim velike i teško predvidive volatilnosti dodatan problem predstavlja kratak period postojanja tog financijskog instrumenta. Većina kriptovaluta

postoji tek nekoliko godina, a neke su čak i nestale samo nekoliko mjeseci nakon što su se pojavile. Period manji od 10 godina je u financijskom svijetu vrlo kratak pogotovo ako uzmemu u obzir da nije bilo prevelikih promjena ili kriza na financijskim tržištima u vremenu postojanja kriptovaluta.

U radu je razvijen C++ programski okvir koji integrira povijesne podatke kriptovaluta, obavlja potrebne matematičke operacije i generira simulacije. Generirane simulacije omogućuju analizu različitih scenarija kretanja cijena, a korisnik može odabrati različite portfelje i vremenske okvire. Sva interakcija s korisnikom odvija se putem grafičkog sučelja koje omogućuje jednostavno upravljanje parametrima simulacije i vizualizaciju rezultata.

Rad je strukturiran kako slijedi: U drugom poglavlju opisuju se teorijske osnove teorije portfelja, Monte Carlo metode, PCA dekompozicija te sve potrebne matematičke osnove i teorije potrebne za razumijevanje i implementaciju modela. Treće poglavlje detaljno opisuje implementaciju algoritama i struktura, uključujući model portfelja, model kriptovalute, postupke poput Cholesky dekompozicije, traženja svojstvene dekompozicije, brzog "parsiranja" financijskog skupa podataka i optimizacije za velike skupove podataka. U četvrtom poglavlju analiziraju se rezultati simulacija za različite konfiguracije portfelja, dok se u zaključku raspravlja o primjenjivosti modela, mogućim i očitim praktičnim problemima i smjerovima daljnog istraživanja.

Ovakav rad može poslužiti kao osnova za daljnje istraživanje i razvoj naprednijih modela i aplikacija koji će omogućiti bolje razumijevanje i upravljanje rizicima povezanim s kriptovalutama, ali i ostalim financijskim instrumentima. Pogotovo jer se kroz izradu sustava i aplikacije kao sekundarni rezultat pokrenuo razvitak biblioteke za rad s financijskim podatcima i potrebnim matematičkim okvirima koja će se moći koristiti u budućim radovima i projektima.

2. Teorija portfelja i matematičke osnove

Teorija portfelja, čiji su začetnici Harry Markowitz i James Tobin, daje strogu matematičku definiciju financijskim pojmovima. Ključni optimizacijski problem teorije portfelja je *dualni cilj*: maksimizacija očekivanog povrata uz istovremeno minimiziranje rizika.

2.1. Investicije

Investicija ili ulaganje je proces alokacije ograničenih resursa kao što su novac, vrijeme ili energija u svrhu ostvarivanja većeg povrata ili dodatne koristi u budućnosti [1]. Ovime kažemo kako investicija ne mora nužno biti financijska, ali u ovom radu se fokusiramo na financijske investicije koje nam predstavljaju izdvajanje i ulaganje novca u različite financijske instrumente kroz određeni vremenski period s ciljem ostvarivanja većeg povrata u budućnosti ili smanjenje gubitka vrijednosti novca.

2.2. Portfelj

Investicijske portfelje matematički prikazujemo kao linearnu kombinaciju pojedinih investicija sa vektorom pojedinih udjela \mathbf{w} .

Definicija 1. Vektor \mathbf{w} predstavlja udjele investicija u portfelju.

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{pmatrix}, \quad \sum_{i=1}^N w_i = 1$$

2.3. Povrati

Povrat investicije je osnovna mjera uspješnosti investicije.

2.3.1. Aritmetički povrat

Definicija 2. Neka je P_t cijena financijskog instrumenta u trenutku t te P_{t-1} cijena istog instrumenta u trenutku $t - 1$. Aritmetički povrat R_t definiramo kao:

$$R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{\Delta P}{P_{t-1}}$$

Neka buduća cijena nam neće biti poznata pri investiranju te zato povrat promatramo kao slučajnu varijablu. Vidimo kako je moguće imati negativan povrat ako je cijena koju promatramo manja od početne cijene i to je upravo situacija koji pokušavamo izbjegći.

2.3.2. Logaritamski povrat

Logaritamski povrat r_t definiramo preko prirodnog logaritma omjera cijena.

Definicija 3. Neka je P_t cijena financijskog instrumenta u trenutku t te P_{t-1} cijena istog instrumenta u trenutku $t - 1$. Logaritamski povrat r_t definiramo kao:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) - \ln(P_{t-1})$$

Logaritamski povrat u pravilu koristimo zbog njegovih pogodnih matematičkih svojstava kao što je svojstvo simetrije $\ln(a) = -\ln(1/a)$ te svojstvo aditivnosti $r_{0,T} = \sum_{t=1}^T r_t$.

2.3.3. Očekivani povrat

Definicija 4. Očekivani povrat promatramo kao srednju vrijednost prijašnjih povrata jer je upravo srednja vrijednost nepristran procjenitelj očekivanja slučajne varijable R_t za koji vrijedi:

$$E(R_t) = \frac{1}{N} \sum_{i=1}^N R_i$$

Definicija 5. Očekivani povrat portfelja je linearna kombinacija očekivanih povrata pojedinačnih asseta:

$$\mathbb{E}[R_p] = \mathbf{w}^\top \boldsymbol{\mu} = \sum_{i=1}^n w_i \mu_i$$

gdje je $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)^\top$ vektor očekivanih povrata.

2.4. Volatilnost

Drugi dio optimizacijskog problema teorije portfelja je smanjenje rizika. Volatilnost je upravo jednostavna mjera rizika koja ima pogodna matematička svojstva. Promatramo je kao standardnu devijaciju slučajne varijable R_t , a ima je smisla tako promatrati jer će nam upravo takva mjera kvantificirati kretanje povrata.

Definicija 6. Volatilnost investicije definiramo kao nepristran procjenitelj standardne devijacije slučajne varijable R_t :

$$\sigma_R = \sqrt{\frac{1}{N-1} \sum_{i=1}^N [R_i - E(R_t)]^2}$$

Definicija 7. Volatilnost portfelja mjeri se standardnom devijacijom povrata i dana je kvadratnim korijenom varijance:

$$\sigma_p = \sqrt{\mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w}}$$

gdje je $\boldsymbol{\Sigma}$ matrica kovarijance s elementima $\Sigma_{ij} = \text{Cov}(r_i, r_j)$.

2.5. Geometrijsko Brownovo gibanje

Geometrijsko Brownovo gibanje (GBM) je jedan od standardnih stohastičkih procesa za modeliranje kretanja cijena financijskih instrumenata. Diferencijalna jednadžba GBM-a

je:

$$dS_t = \mu S_t dt + \sigma S_t dW(t)$$

gdje je $W(t)$ Wienerov proces.

Eksplisitno rješenje GBM-a daje formulu za cijenu u trenutku t :

$$S_t = S_0 \exp \left[\left(\mu - \frac{\sigma^2}{2} \right) t + \sigma W_t \right]$$

gdje je S_0 početna cijena, μ drift, σ volatilnost i W_t Wienerov proces.

Detaljno objašnjene GBM-a i njegovih svojstava te kompletan derivacijski postupak možete pronaći u [2].

2.6. Monte Carlo simulacije

Monte Carlo metoda je numerička metoda koja koristi slučajno uzorkovanje za rješavanje problema. Jedan iznimno intuitivan i elegantan primjer je određivanje vrijednosti broja π . Ideja je generiranje što većeg broja točaka unutar jediničnog kvadrata koji u sebi sadrži jedinični krug te određivanje omjera broja točaka unutar kruga i ukupnog broja točaka te onda preko omjera površina kvadrata i kruga dobijemo procjenu vrijednosti broja π .

U kontekstu financija, Monte Carlo simulacije koriste se za generiranje vjerojatnosnih scenarija budućih cijena. Za portfelj od n instrumenata, koraci su:

1. Generiraj nezavisne slučajne varijable $Z_i \sim N(0, 1)$ (i.i.d.)
2. Transformiraj slučajni vektor nezavisnih varijabli Z u slučajni vektor koreliranih varijabli $\mathbf{Y} = L\mathbf{Z}$ gdje je L donje trokutasta matrica Cholesky dekompozicije Σ
3. Ažuriraj cijene simulacije po GBM formuli za svaki instrument:

$$S_t^{(i)} = S_0^{(i)} \exp \left(\left(\mu_i - \frac{\sigma_i^2}{2} \right) \Delta t + \sigma_i Y_i \sqrt{\Delta t} \right)$$

4. Izračunaj vrijednost portfelja $V_t = \sum_{i=1}^n w_i S_t^{(i)}$

2.7. Cholesky dekompozicija

Cholesky dekompozicija je numerička metoda koja se koristi za dekompoziciju simetričnih pozitivno definitnih matrica. I upravo je matrica kovarijance Σ simetrična pozitivno definitna matrica. Teorem u [3].

Definicija 8. Neka je Σ simetrična pozitivno definitna matrica kovarijance. Tada postoji jedinstvena donja trokutasta matrica L takva da:

$$\Sigma = LL^\top$$

gdje je L donja trokutasta matrica.

Ova faktorizacija omogućuje generiranje vektora koreliranih slučajnih varijabli iz vektora nezavisnih slučajnih varijabli.

Teorem 1. Neka je $\mathbf{Z} = (Z_1, \dots, Z_n)^\top$ vektor nezavisnih $N(0, 1)$ varijabli. Tada vektor $\mathbf{Y} = L\mathbf{Z}$ ima kovarijacijsku matricu Σ .

Dokaz 1.

$$Cov(\mathbf{Y}) = \mathbb{E}[L\mathbf{Z}(L\mathbf{Z})^\top] = L\mathbb{E}[\mathbf{Z}\mathbf{Z}^\top]L^\top = L \cdot I \cdot L^\top = LL^\top = \Sigma$$

2.8. PCA dekompozicija

PCA (Principal Component Analysis) je tehnika koja se koristi za redukciju dimenzionalnosti podataka i identifikaciju glavnih komponenti koje objašnjavaju što veći dio ukupne varijance podataka.

Definicija 9. Neka je Σ kovarijacijska matrica slučajnog vektora $\mathbf{X} = (X_1, X_2, \dots, X_n)^\top$. Svojstvena dekompozicija matrice Σ daje:

$$\Sigma = \mathbf{V}\Lambda\mathbf{V}^\top$$

gdje je \mathbf{V} ortogonalna matrica vlastitih vektora, a $\mathbf{\Lambda}$ dijagonalna matrica vlastitih vrijednosti.

Ovaj rezultat sljedi iz spektralnog teorema [4]. Sada možemo definirati PCA dekompoziciju.

Definicija 10. *PCA dekompozicija slučajnog vektora \mathbf{X} daje novi vektor $\mathbf{Y} = (Y_1, Y_2, \dots, Y_n)^\top$ gdje su Y_i glavne komponente slučajnog vektora \mathbf{X} .*

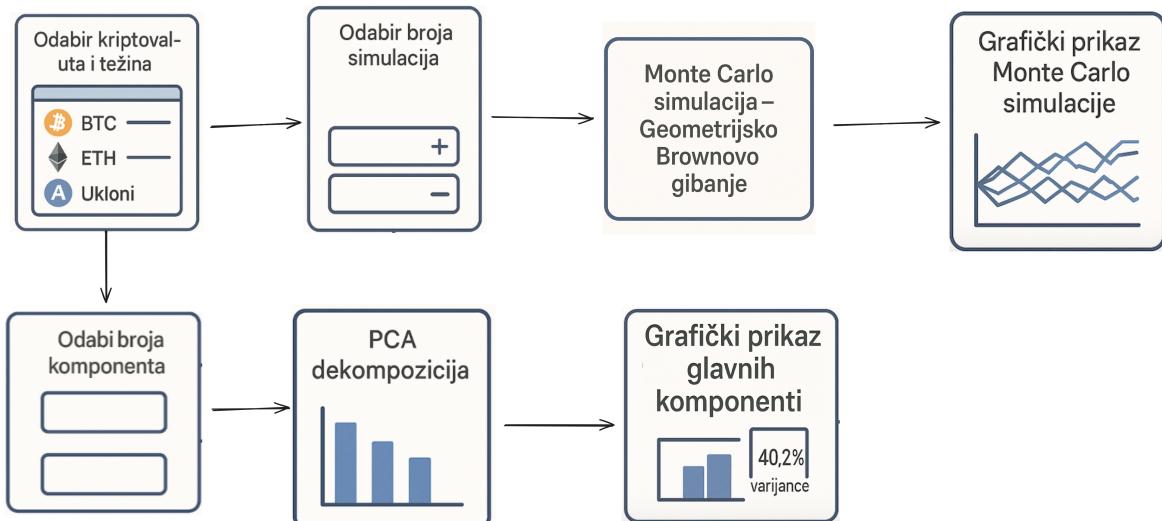
Glavne komponente su linearne kombinacije originalnih varijabli \mathbf{X} i definiraju se kao:

$$\mathbf{Y} = \mathbf{V}^\top \mathbf{X}$$

gdje je \mathbf{V} matrica vlastitih vektora kovarijacijske matrice $\mathbf{\Sigma}$, a Y_i su glavne komponente.

3. Implementacija aplikacije

Ovaj dio rada opisuje implementaciju aplikacije za simulaciju portfelja kriptovaluta. Aplikacija je napisana u C++ programskom jeziku i koristi isključivo standardnu biblioteku C++ za matematičke operacije i modeliranje. Za interakciju s korisnikom koristi se grafičko sučelje(implementirano sa Qt bibliotekom za C++) koje omogućuje vizualizaciju rezultata. Prikaz modela rada aplikacije je dan na slici 3.1.



Slika 3.1. Model rada aplikacije

3.1. Podatci

Podatci korišteni u aplikaciji su povjesne vrijednosti kriptovaluta preuzete sa stranice [5]. Podatci su u CSV formatu i sadrže informacije o otvorenoj, zatvorenoj, najvišoj i najnižoj cijeni te volumenu kriptovalute i ukupnom tržišnom kapitalu za svaki vremenski interval. Za potrebe aplikacije, ovaka pristup je dovoljan jer se fokusiramo na analizu i implementaciju modela portfelja i potrebnih matematičkih operacija. U budućnosti bi

se mogla implementirati i dohvaćanje podataka u realnom vremenu preko nekog javno dostupnog API-ja. Time bi aplikacija postala korisna i za praćenje portfelja u realnom vremenu.

U trenutnim podatcima nalaze se povijesne vrijednosti za 23 različite kriptovalute.

- Aave (AAVE)
- Bitcoin (BTC)
- Ethereum (ETH)
- Binance Coin (BNB)
- Crypto.com Coin (CRO)
- Cardano (ADA)
- Solana (SOL)
- Ripple (XRP)
- Polkadot (DOT)
- Dogecoin (DOGE)
- Litecoin (LTC)
- Chainlink (LINK)
- Uniswap (UNI)
- Bitcoin Wrapped (WBTC)
- Stellar (XLM)
- TRON (TRX)
- Cosmos (ATOM)
- Monero (XMR)
- Thether (USDT)

- USD Coin (USDC)
- Eos (EOS)
- Iota (IOTA)
- NEM (XEM)

3.2. Algoritmi i komponente

Aplikacija se sastoji od nekoliko ključnih komponenti:

- **Parsiranje podataka (3.2.1.)**: učitavanje i obrada povijesnih podataka o cijenama kriptovaluta.
- **Model kriptovaluta (3.2.2.)**: Implementacija modela kriptovaluta.
- **Model portfelja (3.2.3.)**: Definiranje portfelja kriptovaluta.
- **Matematički okvir (3.2.4.)**: Implementacija matematičkih operacija potrebnih za simulaciju i modeliranje portfelja. Npr. (Množenje matrica, vektora, operacije nad vektorima i matricama te razne matrične dekompozicije).
- **Monte Carlo simulacija (3.2.5.)**: Generiranje simulacija budućih cijena portfelja kriptovaluta.
- **PCA dekompozicija (3.2.6.)**: Implementacija PCA dekompozicije za redukciju dimenzionalnosti podataka i identifikaciju glavnih komponenti.
- **Vizualizacija rezultata (3.2.7.)**: Prikaz rezultata simulacija kroz grafičko sučelje.

3.2.1. Parsiranje podataka

Efikasno parsiranje povijesnih podataka o cijenama kriptovaluta je prvi korak u simulaciji i stavarjanju modela portfelja. Podatci se učitavaju iz CSV datoteka koje sadrže povijesne cijene kriptovaluta. Povijesni podatci su preuzeti sa [6]. Podatci se parsiraju u *candle* strukturu koja sadrži otvorenu, zatvorenu, najvišu i najnižu cijenu te volumen kriptovalute za svaki vremenski interval. Također se pri parsiranju računaju i logaritamski povrati za

svaki vremenski interval te se izvršava poravnanje vremenskih oznaka svih kriptovaluta. U pravilu se kriptovaluta može i trguje 24/7, ali je problem bio što neke kriptovalute nisu postojale ili nisu imale podatke za sve dane svih ostalih kriptovaluta.

Kod 3.2.1: Struktura *candle* koja sadrži povijesne podatke o cijenama kriptovaluta

```
1 struct Candle {
2     public:
3         // OHLC
4         double open, high, low, close;
5         double volume, marketcap;
6         timestamp time;
7         double log_return;
8
9     // Constructors, getters, setters, etc.
10};
```

3.2.2. Model kriptovalute

Model kriptovalute je implementiran kao klasa koja sadrži povijesne podatke o cijenama kriptovalute i metode za izračunavanje povrata, volatilnosti i driftova.

Kod 3.2.2: Model kriptovalute

```
1 class Cryptocurrency {
2     public:
3         std::string name, tick;
4         std::vector<Candle> hist_data;
5         bool metrics_calculated = false;
6
7         // Drift = sum(return_i) / n;
8         // Volatility = sqrt((1/n - 1) * sum((return_i - Drift)^2)
9         double drift, volatility;
10
11    // Constructors, getters, setters, etc.
12
13    void calculate_metrics();
```

```

14     void reevaluate_metrics();
15
16     void individual_monte_carlo(int sim_count, int
17         forecast_days);
18
19 }

```

3.2.3. Model portfelja

Model portfelja je implementiran kao klasa koja sadrži vektor kriptovaluta i njihove udjele u portfelju.

Kod 3.2.3: Model portfelja

```

1 class Portfolio {
2
3     private:
4
5         std::vector<std::string> _asset_names;
6         std::unordered_map<Cryptocurrency, double,
7             Cryptocurrency::Hash> _assets;
8
9
10    public:
11
12        // Covariance matrix
13
14        Doubles_Matrix aligned_log_return_matrix;
15
16        Doubles_Matrix covariance_matrix;
17
18        std::vector<double> aligned_means;
19
20        std::vector<double> aligned_volatilities;
21
22
23        timestamp aligned_start = timestamp();
24
25        timestamp aligned_end = timestamp();
26
27        timestamp aligned_stamp = timestamp();
28
29        bool aligned_returns_calculated = false;
30
31        bool covariance_matrix_calculated = false;
32
33        bool aligned_metrics_calculated = false;
34
35
36        void add_asset(const Cryptocurrency &crypto, double
37            ammount);
38
39
40        void remove_asset(const Cryptocurrency &crypto, double
41            ammount);

```

```

    ammount);

23

24     Cryptocurrency get_asset(const std::string &name) const;

25

26     std::vector<std::string> &get_asset_names();

27

28     Doubles_Matrix &aligned_log_returns(timestamp start);

29     Doubles_Matrix &calculate_covariance(timestamp start);

30     int calculate_aligned_metrics(timestamp start);

31     std::vector<Doubles_Matrix> monte_carlo(int simulations,
32
33         int steps,
34
35             timestamp start);

36     int PCA(timestamp start, int num_components,
37
38         std::vector<std::pair<double, std::vector<double>>>
39
40             &components,
41
42             double &total_variance, double &variance_explained);
43
44 };

```

3.2.4. Matematički okvir

Matematički okvir aplikacije implementira osnovne matematičke operacije nad vektorima i matricama, kao što su množenje matrica i vektora te razne matrične dekompozicije. Sve metode koriste samo standardnu biblioteku C++.

Kod 3.2.4: Metode implementirane u matematičkom okviru

```

1 template <typename T>
2 std::vector<std::vector<T>>
3 matrix_transpose(const std::vector<std::vector<T>> &matrix);
4
5 template <typename T>
6 int cholesky_decomposition(const std::vector<std::vector<T>>
7
8     &matrix,
9
10             std::vector<std::vector<T>> &L);
11
12 template <typename T> double vector_norm(const std::vector<T>
13
14     &v);

```

```

9
10 template <typename T> std::vector<T> normalized(const
11   std::vector<T> &v);
12
13 template <typename T>
14 std::vector<T> scalar_multiply(const std::vector<T> &v, T
15   scalar);
16
17 template <typename T>
18 std::vector<std::vector<T>>
19 matrix_scalar_multiply(const std::vector<std::vector<T>>
20   &matrix, T scalar);
21
22 template <typename T>
23 std::vector<T> vector_add(const std::vector<T> &v1, const
24   std::vector<T> &v2);
25
26 template <typename T>
27 std::vector<T> vector_subtract(const std::vector<T> &v1,
28                               const std::vector<T> &v2);
29
30 template <typename T>
31 std::vector<std::vector<T>>
32 matrix_multiply(const std::vector<std::vector<T>> &A,
33                  const std::vector<std::vector<T>> &B);
34
35 template <typename T>
36 std::vector<std::vector<T>>
37 matrix_subtract(const std::vector<std::vector<T>> &A,
38                  const std::vector<std::vector<T>> &B);

```

```

39
40
41
42 template <typename T>
43 int QR_decomposition(const std::vector<std::vector<T>> &matrix ,
44                     std::vector<std::vector<T>> &Q ,
45                     std::vector<std::vector<T>> &R);
46
47 template <typename T> int normalize(std::vector<T> &v);
48
49 template <typename T>
50 int matrix_inverse(const std::vector<std::vector<T>> &input ,
51                     std::vector<std::vector<T>> &output) {
52
53 template <typename T>
54 std::vector<T> eigen_values(const std::vector<std::vector<T>>
55 &matrix);
56
56 template <typename T>
57 int eigen_pairs(const std::vector<std::vector<T>> &matrix ,
58                 std::vector<std::pair<T, std::vector<T>>>
59                 &results);

```

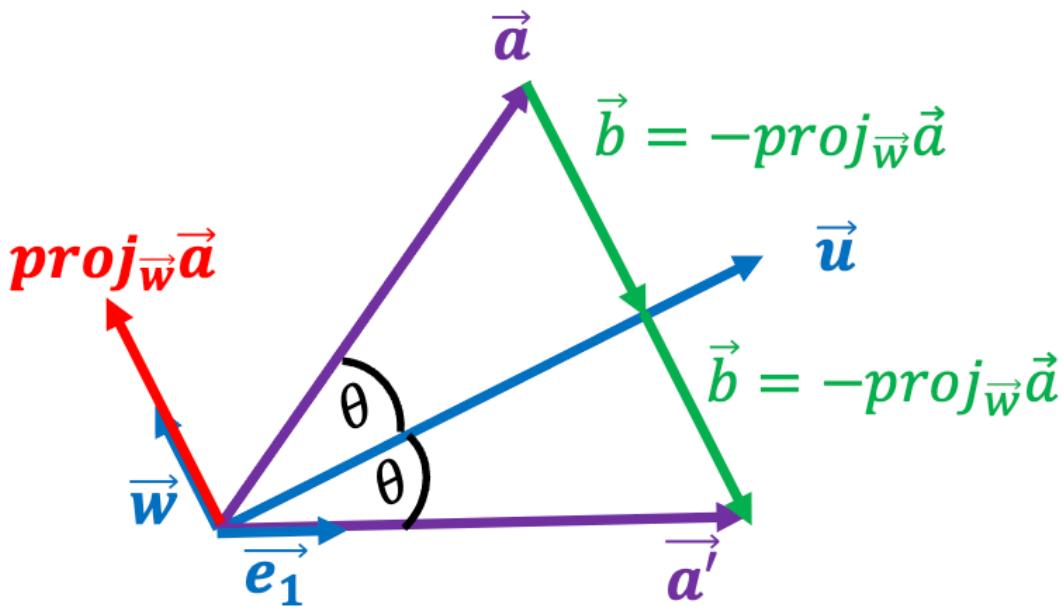
Cholesky dekompozicija

Cholesky dekompozicija je implementirana kao metoda koja prima kovarijacijsku matricu i matricu L u koju se spremaju donja trokutasta matrica dekompozicije te metoda vraća int koji označava uspješnost dekompozicije. Algoritam radi točno prema koracima matematične notacije. Znamo da je matrica kovarijance uvijek simetrična i pozitivno definitna, pa je Cholesky dekompozicija uvijek moguća.

QR dekompozicija

QR dekompozicija je implementirana kao metoda koja prima ulaznu matricu te matrice Q i R u koje se spremaju rezultati dekompozicije. Implementacija QR dekompozicije koristi

Householderove refleksije za dekompoziciju matrice. Ideja Householderove refleksije je pronaći refleksiju koja će transformirati prvi stupac matrice u vektor $\begin{pmatrix} \|x\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. Nakon što se prvi stupac transformira, iterativno se primjenjuju Householderove refleksije na preostale stupce matrice.



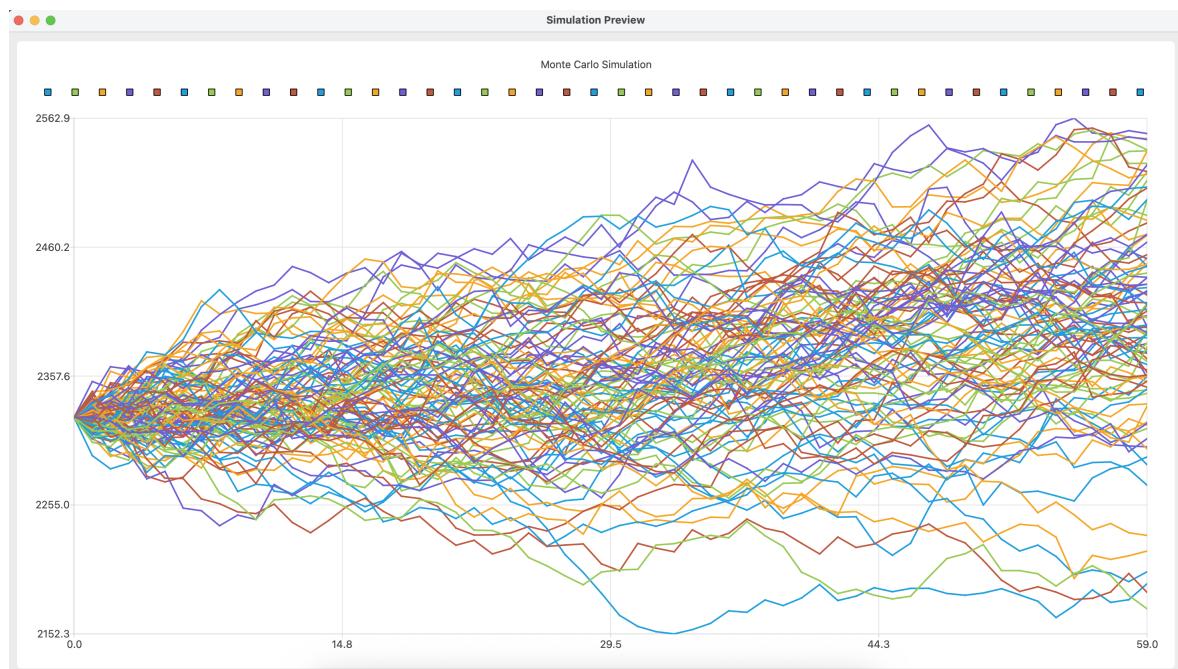
Slika 3.2. Primjer Householderove refleksije, preuzeto sa [7]

Svojstvena dekompozicija

Svojstvena dekompozicija je implementirana kao metoda koja prima kovarijacijsku matricu te par vlastitih vrijednosti i vlastitih vektora u koje se spremaju rezultati dekompozicije. Za izračun vlastitih vrijednosti koristi se *metoda QR iteracija* [8] koja QR dekompoziciju iterativno primjenjuje na matricu dok ne dobijemo i onda množi matricu R i Q te tako dobivamo novu matricu koja je približno dijagonalna. Zatim se vlastiti vektori dobivaju posebno za svaku vlastitu vrijednost koristeći *metodu inverznih iteracija* [8]. Ovakva implementacija nije najefikasnija i sigurno među prvim optimizacijama koje bi se mogle i trebale napraviti.

3.2.5. Monte Carlo simulacija

Monte Carlo simulacija je implementirana kao metoda klase `Portfolio` koja generira simulacije budućih cijena portfelja kriptovaluta. Simulacije koriste model geometrijskog Brownovog gibanja za generiranje budućih cijena te Cholesky dekompoziciju za osiguranje uzoračke korelacije između kriptovaluta. Implementacija Monte Carlo simulacija prvo koristi metodu za izračuvanje kovarijacijske matrice portfelja iz povijesnih podataka kriptovaluta. Također se može postaviti početni datum simulacije, broj simulacija i broj koraka simulacije. Zatim se koristeći `std::normal_distribution(0, 1)` generiraju nezavisne slučajne varijable ($\sim N(0, 1)$) koje se transformiraju u korelirane slučajne varijable pomoću Cholesky dekompozicije. Zatim se ažuriraju cijene simulacije koristeći geometrijsko Brownovo gibanje prema formuli: 2.5. Primjer rezultata simulacije je dan na slici 3.3.

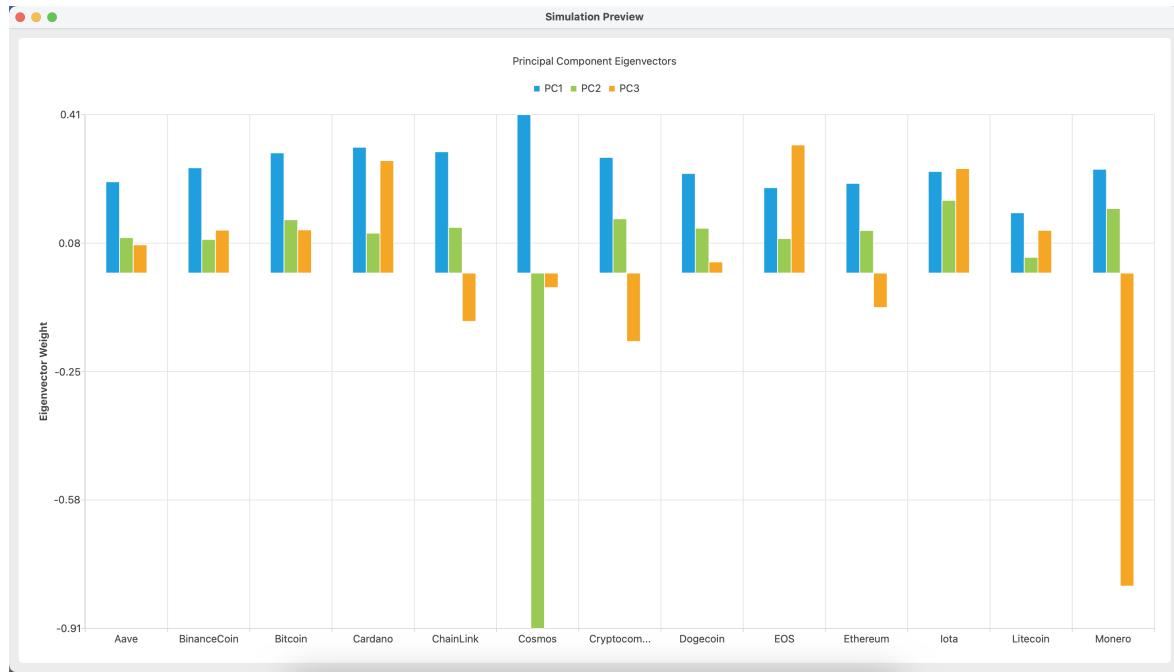


Slika 3.3. Primjer Monte Carlo simulacije portfelja kriptovaluta (100 simulacija, 60 dana)

3.2.6. PCA dekompozicija

PCA dekompozicija je također implementirana kao metoda klase `Portfolio` koja prima ulaznu kovarijacijsku matricu, broj glavnih komponenti i vektor u koji se spremaju glavne komponente poredane po količini varijance koju objašnjavaju te ukupnu varijancu (u ovom slučaju to je suma svih vlastitih vrijednosti) i varijancu objašnjenu glavnim

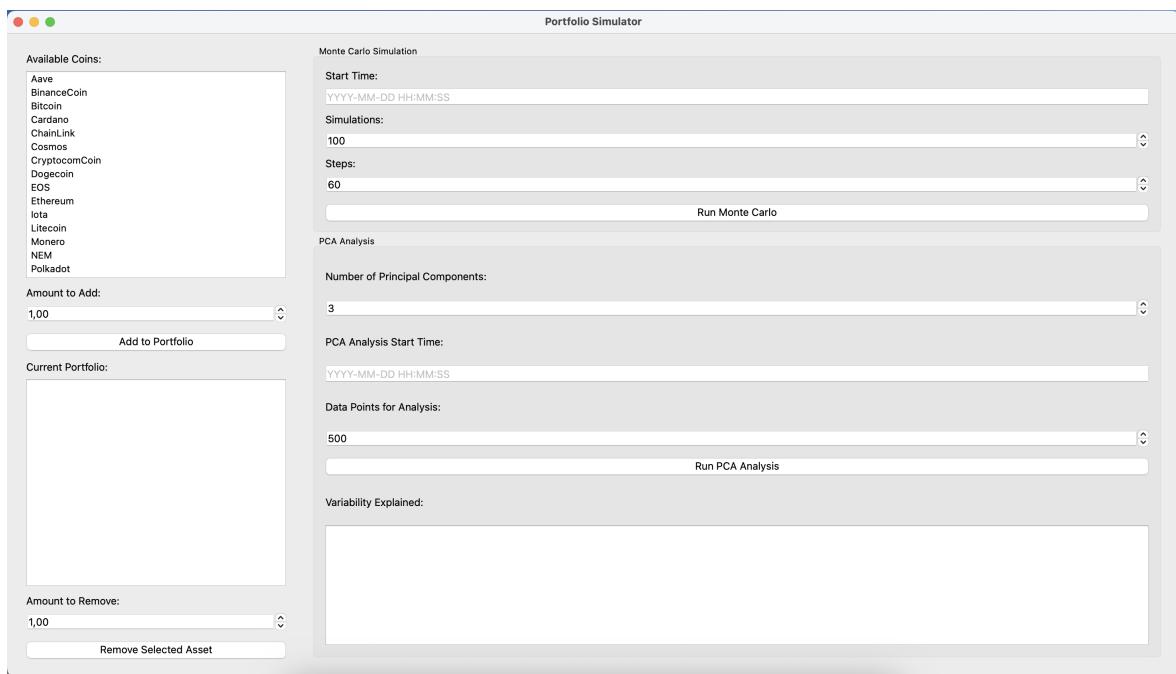
komponentama. Metoda vrati int koji označava uspješnost dekompozicije. Za izračun glavnih komponenti koristi se svojstvena dekompozicija koja daje vlastite vrijednosti i vlastite vektore kovarijacijske matrice te je objašnjena u 3.2.4. primjer rezultata PCA dekompozicije je dan na slici 3.4.



Slika 3.4. Primjer PCA dekompozicije portfelja kriptovaluta (3 glavne komponente)

3.2.7. Vizualizacija rezultata

Za vizualizaciju rezultata i interakciju s korisnikom koristi se grafičko sučelje implementirano u C++ koristeći Qt biblioteku. Grafičko sučelje omogućuje korisniku da odabere kriptovalute koje želi staviti u portfelj, postavi udjele kriptovaluta u portfelju, odabere vremenski period simulacije i broj simulacija. Rezultati simulacija se prikazuju u obliku grafova koji se otvaraju u novom prozoru aplikacije i prikazuju teoretsko kretanje vrijednosti portfelja u budućnosti. Korisnik također može odabrati broj glavnih komponenti te izvrši PCA dekompoziciju portfelja. Prikaz rezultata PCA dekompozicije je također u obliku grafova koji se otvara u novom prozoru aplikacije i prikazuje glavne komponente portfelja te varijancu koju objašnjavaju.



Slika 3.5. Grafičko sučelje aplikacije za simulaciju portfelja kriptovaluta

4. Rezultati i rasprava

U ovom poglavlju prikazujemo rezultate postignute implementacijom aplikacije te raspravljamo o njihovoj važnosti i primjenjivosti u stvarnom svijetu. Također, analiziramo prednosti i nedostatke implementacije te mogućnosti optimizacija i proširenja aplikacije.

4.1. Implementacijski rezultati

4.1.1. Funkcionalnost učitavanja i parsiranja podataka

Ručna implementacija parsiranja podataka omogućila je optimizaciju učitavanja i obrade podataka zbog specifičnosti izgleda samog skupa podataka. Osim što odmah znamo koji su stupci prisutni dodatna optimizacija je postignuta pregledavanjem veličine datoteke i aproksimacijom broja redaka u datoteci. To nam omogućuje da unaprijed rezerviramo memoriju za vektor *candle* struktura što značajno ubrzava proces parsiranja pogotovo kod većih skupova podataka jer ne moramo konstantno realocirati memoriju pri proširivanju vektora. Ova implementacija je na dostupnim skupovima podataka postigla brzinu parsiranja usporedivu s popularnom bibliotekom *csv2* [9] koja je optimizirana za parsiranje CSV datoteka.

4.1.2. Matematički okvir

Matematički okvir aplikacije implementiran je tako da se koristi samo standardna biblioteka C++. Iako je ovakav pristup produljio vrijeme izrade aplikacije te samo izvođenje aplikacije nije optimizirano kao što bi bilo da se koriste biblioteke poput *Eigen* [10] ili *Armadillo* [11], koje koriste optimizirane algoritme i strukture podataka za matematičke operacije te napredne optimizacije poput SIMD instrukcija, template metaprogramming-a i expression templates-a, ovakav pristup je omogućio dublje razumijevanje i primjenu stečenih znanja.

Implementacija se specifično oslanja na `std::vector` kao osnovnu strukturu podataka za vektore i matrice. `std::vector` je dinamički niz koji omogućuje efikasno upravljanje memorijom i automatsko proširivanje kada je potrebno.

Većina matematičkih operacija je implementirana na *naivni* način kako bi se osiguralo da su svi koraci jasni i razumljivi. Jedine optimizacije su napravljene koristeći *move semantics* i kako bi se izbjeglo nepotrebno kopiranje podataka te korištenje OpenMP-a [12] za paralelizaciju određenih operacija. Sve operacije su testirane na skupnjениm skupovima podataka i vrijeme izvođenja je zadovoljavajuće za većinu operacija. Jedino što je potrebno optimizirati je traženje vlastitih vektora koristeći neki od asimptotski efikasnijih algoritama u odnosu na *metodu inverznih iteracija*.

4.1.3. Monte Carlo simulacija

Monte Carlo simulacija se pokazala kao efikasan način generiranja vjerojatnosnih scenarija budućih cijena portfelja kriptovaluta ukoliko je inicijalni uzorak smislen i dovoljno velik. Jedan primjer simulacije koji izgleda realistično i smislen je dan je ranije na slici 3.3. Na tom se primjeru vidi blagi pozitivni *drift* cijena portfelja te volatilnost koja je u skladu s očekivanjima za portfelj kriptovaluta.

Drugi primjer simulacije je dan na slici ??

Figures/monte_carlo_example2.png

Slika 4.1. Primjer Monte Carlo simulacije portfelja kriptovaluta (100 simulacija, 60 dana)

5. Zaključak

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Literatura

- [1] Z. Bodie, A. Kane, i A. J. Marcus, *Investments*. McGraw-Hill, 2014.
- [2] J. Campana i C. Grenon, “Deriving the black-scholes formula from brownian motion”, 2021.
- [3] J. P. Milišić i A. Žgaljić Keko, *Uvod u numeričku matematiku za inženjere*. Element, 2014.
- [4] A. A. Aljinović, N. Elezović, i D. Žubrinić, *Linearna algebra*. Element, 2017.
- [5] S. Rajkumar, “Cryptocurrency historical prices”, <https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory/data>.
- [6] K. L. Pavasović, “Metode neglatke optimizacije u teoriji portfelja”, Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, 2022.
- [7] A. Kwok, “Detailed explanation of qr decomposition by householder transformation”, <https://kwokanthony.medium.com/detailed-explanation-with-example-on-qr-decomposition-by-householder-transformation-5e964d7f7656>.
- [8] N. Truhar, *Numerička linearna algebra*. Odjel za matematiku, Svučilište J. J. Strossmayera u Osijeku, 2010.
- [9] “Csv2”, <https://github.com/p-ranav/csv2>.
- [10] G. Guennebaud, B. Jacob *et al.*, “Eigen v3”, <http://eigen.tuxfamily.org>, 2010.
- [11] C. Sanderson i R. Curtin, “Armadillo: an efficient framework for numerical linear algebra”, u *2025 17th International Conference on Computer and Automation*

Engineering (ICCAE), 2025., str. 303–307. <https://doi.org/10.1109/ICCAE64891.2025.10980539>

- [12] L. Dagum i R. Menon, “Openmp: an industry standard api for shared-memory programming”, *IEEE Computational Science and Engineering*, sv. 5, br. 1, str. 46–55, 1998. <https://doi.org/10.1109/99.660313>

Sažetak

Razvoj aplikacije za procjenu rizika u investicijskim portfeljima uz pomoć Monte Carlo simulacija

Ivan Džanija

Unesite sažetak na hrvatskom.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Ključne riječi: prva ključna riječ; druga ključna riječ; treća ključna riječ

Abstract

Development of an application for risk assessment in investment portfolios with the help of Monte Carlo simulations

Ivan Džanija

Enter the abstract in English.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Keywords: the first keyword; the second keyword; the third keyword

Privitak A: The Code

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam

rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.