On-line Monte Carlo simulacija

Ivanković, Margita

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:681962

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Margita Ivanković

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Prof. dr. sc. Mario Essert, dipl. ing. Margita Ivanković

Zagreb, 2017.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se prof. dr. sc. Mariu Essertu na mentorstvu i pristupačnosti tijekom pisanja rada.

Margita Ivanković

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Margita Ivanković** Mat. br.: 0035196510

Naslov rada na hrvatskom jeziku: **On-line Monte Carlo simulacija**

Naslov rada na engleskom jeziku: **On-line Monte Carlo simulation**

Opis zadatka:

Monte Carlo simulacija je niz računalnih eksperimenata koja se svodi na ponavljanje slučajnog uzorkovanja kako bi se dobili numerički rezultati koji u načelu mogu biti determinističkog tipa, ali su, zbog velikog broja podataka, vrlo teško ili gotovo nemoguće rješivi na drugi način. Ta metoda najčešće se provodi u numeričkim integracijama, optimizacijama i distribucijama vjerojatnosti. Usko je povezana sa Markovljevim lancima (npr. MCMC algoritam) za probleme u višedimenzionalnom vektorskom prostoru, gdje se novi podatčani uzorak dobiva na temelju njegovih najbližih susjeda, to jest, tranzicijske vjerojatnosti izmedju uzorkovanih vrijednosti ovise samo o vrijednostima zadnjih uzoraka. Te i takve vrste simulacija temelji su mnogih metoda današnjeg strojnog učenja.

U ovom radu, potrebno je:

1. Upoznati i objasniti matematičku pozadinu Monte Carlo simulacije
2. Proučiti postojeća programska rješenja Monte Carlo simulacija, npr. python modul montecarlo 0.1.17, mcerp 0.11 ili pymc 2.3.6 koja su već načinjena u programskom jeziku Python, ali nažalost nemaju mrežne (Internetske) mogućnosti
3. Preinakom postojećih modula, načiniti Python modul za Monte Carlo simulacije i ugraditi ga u web2py mrežni okvir (eng. framework)
4. U mrežnom okviru načiniti odgovarajuće numeričko i grafičko sučelje za unos podataka i on-line simulaciju
5. Testirati tako dobiveni alat s različitim tehničkim i netehničkim podatcima, preko Interneta

Zadatak zadan:

30. studenog 2016.

Zadatak zadao:

M. Essert

Prof.dr.sc. Mario Essert

Rok predaje rada:
1. rok: 24. veljače 2017.
2. rok (izvanredni): 28. lipnja 2017.
3. rok: 22. rujna 2017.

Predviđeni datumi obrane:
1. rok: 27.2. - 03.03. 2017.
2. rok (izvanredni): 30. 06. 2017.
3. rok: 25.9. - 29. 09. 2017.

v.d. predsjednika Povjerenstva:

Izv. prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS OZNAKA III
SAŽETAK IV
SUMMARY V
1. UVOD 1
2. MONTE CARLO SIMULACIJA 2
2.1. Općenito 2
2.2. Slučajne varijable 2
2.3. Generiranje slučajnih varijabli 4
2.4. Normalna distribucija 5
2.4.1. Pravilo tri sigme 6
2.5. Latin hypercube metoda uzorkovanja 7
3. PRIMJENA MONTE CARLO SIMULACIJE 8
3.1. Statističke tolerancije 8
3.1.1. Slaganje tolerancija 8
3.1.2. Strujni krug 9
3.1.3. Procjenjivanje trajanja projekta 9
4. PYTHON I web2py MREŽNI OKVIR 11
4.1. Python programski jezik 11
4.2. Općenito o web2py 11
4.3. Implementiranje Python modula mcerp u web2py 12
4.3.1. Definiranje funkcije Latin hypercube-a s Monte Carlo metodom 12
4.3.2. Definiranje funkcija distribucije 14
4.3.2.1. Normalna distribucija 16
4.3.2.2. PERT metoda 16
4.3.3. Definiranje vjerojatnosti 16
5. ON-LINE SIMULACIJA 18
5.1. Početna stranica 18
5.2. Stranica Tolerancije 19
5.2.1. Simulacija slaganja tolerancija 19
5.2.2. Simulacija strujnog kruga 22
5.3. Stranica trajanje projekta 24
5.4. Model web aplikacije 24
5.5. Controller web aplikacije 25
5.6. View web aplikacije 28
6. ZAKLJUČAK 29
LITERATURA 30
PRILOZI 31

POPIS SLIKA

Slika 1. Prikaz funkcije gustoće (lijevo) i funkcija distribucije (desno) [5] 5
Slika 2. Normalna distribucija [6] 6
Slika 3. Latin hypercube uzorkovanje [8] 7
Slika 4. Primjer slaganja tolerancija 9
Slika 5. PERT i Beta distribucija [10] 10
Slika 6. Definiranje funkcije lhd 12
Slika 7. Defniranje funkcije _lhs 13
Slika 8. Definiranje funkcije _mix 13
Slika 9. Slučajno generiranje brojeva 14
Slika 10. Definiranje klase NeizvjesnaFunkcija 15
Slika 11. Definiranje klase NeizvjesnaVarijabla 15
Slika 12. Definiranje funkcije Normal 16
Slika 13. Definiranje PERT metode 16
Slika 14. Definiranje funkcija __le__ i __ge__ 17
Slika 15. Početna stranica 18
Slika 16. Stranica Statističke tolerancije 19
Slika 17. Stranica slaganje tolerancija 19
Slika 18. Unos podataka 20
Slika 19. Excel tablica 20
Slika 20. Prikaz rezultata slaganja tolerancija 21
Slika 21. Histogram raspodjele zračnosti 21
Slika 22. Stranica Strujni krug 22
Slika 23. Prikaz rezultata struje i snage 23
Slika 24. Prikaz histograma struje 23
Slika 25. Procjenjivanje trajanja projekta 24
Slika 26. Kreiranje baze podataka 24
Slika 27. Baza podataka „projekt“ 25
Slika 28. Funkcija tolerancije_1 25
Slika 29. Funkcija rezultat1 26
Slika 30. Funkcija graf 26
Slika 31. Funkcija rezultat za iznose struje 27
Slika 32. Naredba onvalidation 27
Slika 33. Prikaz view-a 28

POPIS OZNAKA

Oznaka	Jedinica	Opis
a		Optimistično vrijeme PERT metode
b		Pesimistično vrijeme PERT metode
$E(x)$		Matematičko očekivanje
$g(x)$		Funkcija koja povezuje izlaznu s ulaznom varijablom
$f(x)$		Funkcija gustoće vjerojatnosti
$F(x)$		Kumulativna razdioba ili funkcija distribucije vjerojatnosti
$F^{-1}(x)$		Inverz funkcije distribucije vjerojatnosti
I	mA	Jakost električne struje
$L1,L2,L3,L4$	mm	Srednja vrijednost dimenzija
m		Normalno vrijeme PERT metode
M		Segmenti Latin hypercube uzorkovanja
N		Broj uzoraka (simulacija)
P		Vjerojatnost
R	kΩ	Električni otpor
Te		Očekivano vrijeme PERT metode
U		Uniformna distribucija
V	V	Napon izvora
$V(x)$		Varijanca
X		Slučajna varijabla
Y		Izlazna varijabla
Z	mm	Zračnost
α, β		Parametri Beta distribucije
μ		Srednja vrijednost normalne distribucije
σ		Standardna devijacija normalne distribucije

SAŽETAK

Monte Carlo metoda je numerička metoda rješavanja matematičkih problema slučajnim uzorkovanjem velikog broja varijabli. Koristi se za rješavanje problema koji su prekompliciran za riješiti analitički. U web2py mrežnom okviru je izrađena on-line Monte Carlo simulacija s primjerima vezanim za slaganje tolerancija odnosno dimenzija i utjecaj tolerancija elektroničkih komponenti na strujni krug. Za opis raspodjele tolerancija je korištena normalna distribucija. PERT metoda skupa s Monte Carlom može procjeniti trajanje projekta što je također simulirano u aplikaciji. Monte Carlo simulacija povezana s Latin hypercube uzorkovanjem je korištena u obliku preinačenog mcerp 0.11 Python modula.

Ključne riječi: Monte Carlo, web2py, slučajno uzorkovanje, Latin hypercube, tolerancije, normalna distribucija, PERT, mcerp

SUMMARY

The Monte Carlo method is a numerical method of solving mathematical problems by radnom sampling a large number of variables. It is used for obtaining numerical solutions to problems which are too complicated to solve analytically. On-line Monte Carlo simulation is made in web2py framework with examples based on tolerance stackup and effects of electrical component variatons in electrical circuit. Normal distribution is used to describe tolerance range. PERT method used with Monte Carlo can estimate the time needed to complete given project which is also shown in web application. Monte Carlo method that uses Latin hypercube sampling is used with Python module mcerp 0.11.

Key words: Monte Carlo, web2py, random sampling, Latin hypercube, tolerance, normal distribution, PERT, mcerp

1. UVOD

Monte Carlo metoda je bilo koji matematički model i algoritam čija je glavna značajka korištenje velikog broja slučajnih brojeva u rješavanju različitih problema. Metoda se najčešće koristi za dobivanje numeričkih rješenja kod problema koje je vrlo teško riješiti analitički. Monte Carlo simulacija ulazi u širu upotrebu tek nakon naglog razvoja računala zbog velikog broja ponavljanja i matematičkih operacija. Provodi se najčešće u numeričkim integracijama, distribucijama vjerojatnosti, rizika, ako postoji neizvjesnost u pojavljivanju ulaznih varijabli, ali i mnogim drugim područjima. U strojarstvu se može koristiti za analizu statističkih tolerancija, koje su bazirane na masovnoj proizvodnji. Provedena Monte Carlo simulacija na velikom broju uzoraka pokazuje koliki utjecaj ima određena tolerancija na krajnji proizvod te kolika će biti greška izvan zadanih granica. U web2py mrežnom okviru je napravljena web aplikacija za Monte Carlo simulaciju na primjerima sa slaganjem tolerancija i utjecajem odstupanja elektroničkih komponenti u strujnom krugu. Učitava se postojeća datoteka s podatcima i zatim aplikacija daje rezultate i grafički prikaz rezultata ovisno o odabranom primjeru. Također je načinjeno sučelje za procjenivanje trajanja nekog projekta to jest aktivnosti, koje je isto tako bazirano na Monte Carlo metodi. Za izvedbu Monte Carlo simulacije napravljena je preinaka postojećeg Python modula mcerp 0.11 kojim se definira funkcija ulaznih podataka te slučajno uzorkovanje tih podataka bazirano na Latin hypercube metodi.

2. MONTE CARLO SIMULACIJA

2.1. Općenito

Monte Carlo simulacija radi slučajno uzorkovanje i provodi velik broj eksperimenata, promatraju se statističke karakteristike tih eksperimenata to jest karakteristike izlaznih varijabli zadanog matematičkog modela i izvlače se zaključci iz tih izlaza bazirani na statističkim eksperimentima. U svakom modelu se generiraju slučajne varijable $X=(X_1,X_2,...,X_n)$ te njihove vjerojatnosti ovisno o njihovoj distribuciji. Zatim se računaju vrijednosti izlaznih varijabli koje su opisane funkcijom $Y=g(X)$, veza između izlazne i ulazne varijable. Ovisno o broju uzoraka, toliko će se generirati ulaznih slučajnih varijabli iz zadane domene to jest funkcije željene distribucije i isto toliko će se dobiti izlaznih varijabli. Tijek Monte Carlo simulacije je sljedeći:

1. Definiranje matematičkog modela koji povezuje izlazne s ulaznim varijablama
2. Definiranje distribucije ulaznih varijabli
3. Generiranje slučajnih varijabli iz funkcije definirane distribucije
4. Zadaje se broj simulacija N i treći korak se ponavlja N puta
5. Dobiva se N slučajnih ulaznih varijabli
6. Generiranje N vrijednosti izlazne varijable koja je izračunata iz matematičkog modela
7. Iz N vrijednosti izlaznih varijabli računa se funkcija distribucije izlazne varijable
8. Dobiva se distribucija izlazne varijable i njena vjerojatnost

Monte Carlo simulacija je često usko povezana s Markovljevim lancima. MCMC algoritam znači da je svako buduće stanje vremenski neovisno o svakom prijašnjem stanju odnosno da svaki novi uzorak ovisi samo o vrijednosti zadnjeg uzorka. Također je definiran time da se stacionarna distribucija podudara s ciljanom.

2.2. Slučajne varijable

Svrha uzorkovanja slučajnih ulaznih varijabli je kako bi generirali uzorke koji predstavljaju distribuciju ulaznih podataka. Slučajna varijabla je numerički ishod slučajnog eksperimenta. Ako postoji skup događaja i funkcija X koja događajima iz tog skupa pridružuje različite x_n, funkcija X se zove slučajna varijabla. Slučajna varijabla X je neprekidna (kontinuirana) ako prima svaku vrijednost na nekom intervalu. Svaka slučajna varijabla ima svoju vjerojatnost P. Ako je distribucija vjerojatnosti zadana nekom funkcijom f koja je zadana na čitavom skupu realnih brojeva, za tu funkciju f vrijede jednadžbe (1) i (2) :

$$f(x) \geq 0 \quad \textbf{(1)}$$

$$\int_{-\infty}^{\infty} f(x)dx = 1 \quad \textbf{(2)}$$

Funkcija *f* se tada naziva funkcija gustoće vjerojatnosti, dakle ta funkcija definira neku distribuciju vjerojatnosti neprekidne slučajne varijable.

Vjerojatnost da slučajna varijabla *X*, koja ima funkciju gustoće *f*, poprimi vrijednost na intervalu [*a*,*b*] jednaka je (3):

$$P(a \leq X \leq b) = \int_{a}^{b} f(x)dx \quad \textbf{(3)}$$

Funkcija *F* je kumulativna razdioba ili funkcija distribucije slučajne varijable i izražena je jednadžbom (4):

$$F(a) = \int_{-\infty}^{a} f(x)dx \quad \textbf{(4)}$$

Ako je poznata funkcija distribucije *F*, moguće je izračunati vjerojatnost preko (5):

$$P(a \leq X \leq b) = F(b) - F(a) \quad \textbf{(5)}$$

Očekivanje slučajne varijable je očekivana vrijednost to jest prosjek svih varijabli *X*, i izražava se jednadžbom (6):

$$E(x) = \mu = \int_{-\infty}^{\infty} xf(x)dx \quad \textbf{(6)}$$

Varijanca slučajne varijable je raspršenje *X* od *E(X)*, koja se izražava jednadžbom (7):

$$V(X) = \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx \quad \textbf{(7)}$$

Standardna devijacija ili σ je drugi korijen od varijance slučajne varijable.

2.3. Generiranje slučajnih varijabli

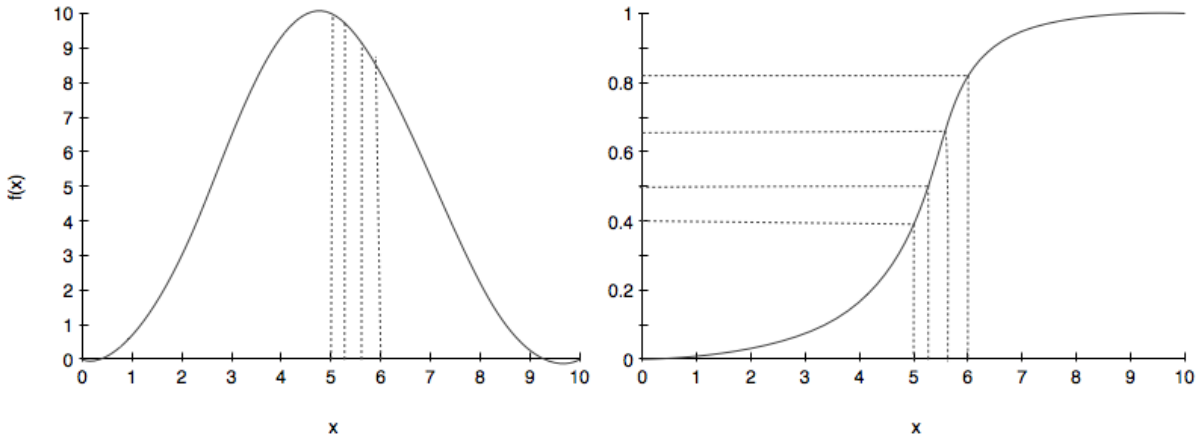
Kako ne postoje istinski slučajni brojevi radi se o pseudoslučajnim brojevima, koji se čine slučajni, ali u suštini nisu. Glavna razlika između njih je što će se generator pseudoslučajnih brojeva nakon nekog vremena početi ponavljati što se kod istinski slučajnih brojeva neće dogoditi. Generatori pseudoslučajnih brojeva su algoritmi koji kao izlaz vraćaju statistički potpuno nezavisne i nepredvidljive vrijednosti u obliku brojevnog niza. U Python programskom jeziku slučajni brojevi se mogu dobiti naredom *random()*, koja koristi Mersenne Twister kao generator. Taj generator daje 53-bitno precizne decimalne brojeve i ima period od 2^{19937}-1. Pseudoslučajni brojevi koji se generiraju su uniformno distribuirani na intervalu [0.0, 1.0). Važnost uniformih brojeva jednoliko i kontinuirano raspoređenih na tom intervalu je da se mogu transformirati u prave varijable koje pripadaju traženoj razdiobi. Najlakša i direktna transformacija je metoda inverzne transformacije. Pretpostavlja se da je X neprekidna slučajna varijabla s funkcijom distribucije $F(x)$, gdje je F neprekidna i rastuća funkcija koja vraća vjerojatnost da će X imati vrijednost manju ili jednaku x (8).

$$F(x) = P(X \leq x) \tag{8}$$

Neka se odabire slučajna varijabla U iz uniformne distribucije $U(0,1)$ i funkcija distribucije F neprekidne slučajne varijable X koja se želi modelirati. Tada se distribucija slučajne varijable X može dobiti kao distribucija kompozicije $F^{-1}(U)$. Gdje vrijedi jednadžba (9):

$$P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x) \tag{9}$$

Dakle, ako je potrebno generirati N podataka iz neprekidne slučajne varijable s distribucijom F, dovoljno je odabrati niz podataka (u_i, i=1,...n) iz $U(0,1)$. I tražene varijable se tada dobiju kao ($F^{-1}(u_i)$, i=1,...n). Funkcija distribucije daje vjerojatnost varijable, a inverz funkcije distribucije vraća vrijednost varijable. Na slici [Slika 1] je prikaz generiranja slučajnih varijabli pomoću funkcije distribucije. Na lijevoj strani slike je graf funkcije gustoće *f(x)*, a na desnoj je njena funkcija distribucije *F(x)* koja na ordinati ima raspodijelu uniformno distribuiranih slučajnih varijabli i na apscisi njihove vrijednosti.



Slika 1. Prikaz funkcije gustoće (lijevo) i funkcija distribucije (desno) [5]

2.4. Normalna distribucija

Normalna distribucija, još poznata kao i Gaussova razdioba, uobičajen je model za prikaz varijacija i nasumičnosti. Najčešće se koristi jer je bit u tome da kada se stvari mijenjaju ipak nastoje ostati u blizini prosječne vrijednosti i raspoređuju se oko tog prosjeka po zvonikolikoj krivulji. Normalna distribucija nije definira na cijelom području, nikad ne presijeca x os, nego joj se samo približava. Proteže se i definirana je samo na određenom intervalu pa ju stoga možemo nazvati neizvjesnom. Parametri normalne distribucije su srednja vrijednost ili očekivanje μ i standardna devijacija σ stoga slučajna varijabla koja ima normalnu distribuciju se označava $X \sim N(\mu, \sigma^2)$. Njena funkcija gustoće je tada dana formulom (10):

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{10}$$

Gdje su μ i σ realni brojevi i $\sigma > 0$.

Funkcija distribucije za normalnu distribuiranu varijablu predstavlja vjerojatnost $P(X \leq x) = F(x)$ da će slučajna varijabla X poprimiti vrijednost koje su jednake ili manje od x. Izražena je jednadžbama (11) i (12):

$$F(x) = \int_{-\infty}^{x} f(x)dx \tag{11}$$

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \quad \textbf{(12)}$$

2.4.1. Pravilo tri sigme

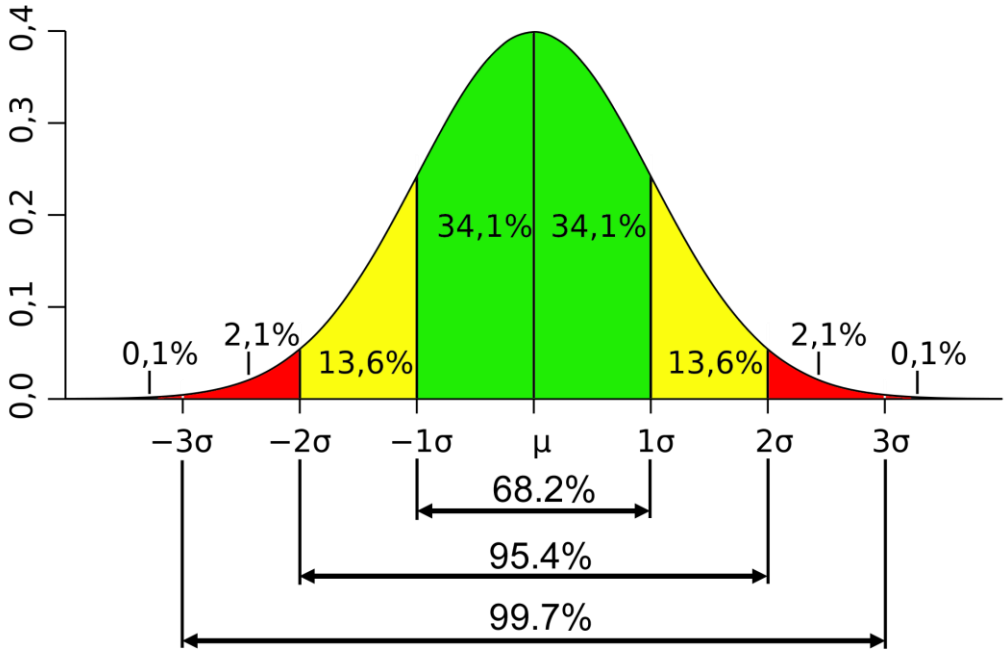
Vjerojatnosti varijabli koje su na intervalima oko srednje vrijednosti na širini od dvije, četiri ili šest standardnih devijacija se mogu prikazati jednadžbama :

$$P(\mu - \sigma \leq X \leq \mu + \sigma) \approx 0.6827 \quad \textbf{(13)}$$

$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 0.9545 \quad \textbf{(14)}$$

$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 0.9973 \quad \textbf{(15)}$$

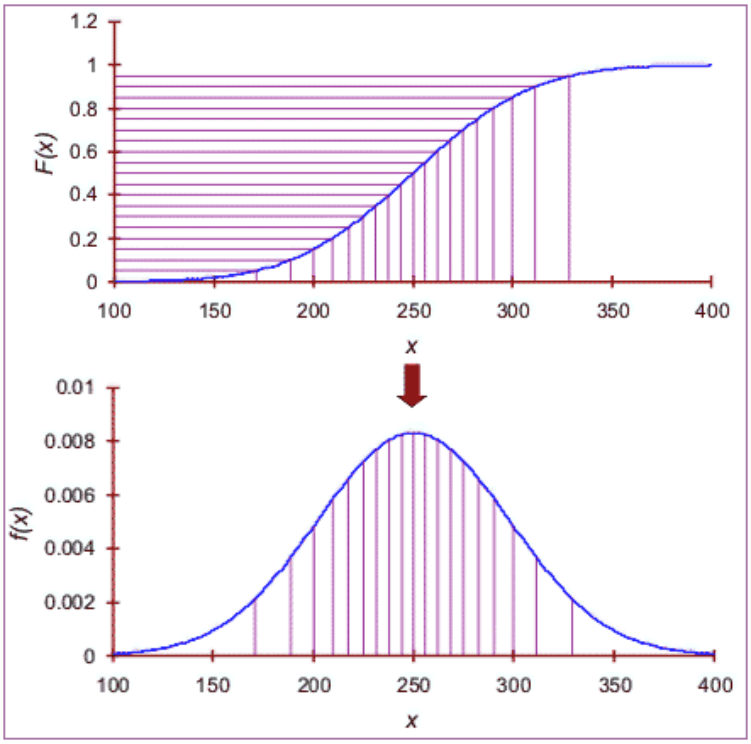
Vjerojatnost od 0.9973 znači da će 99.73% vrijednosti neprekidnih slučajnih varijabli pripadati intervalu od -3σ do +3σ. Ta karakteristika se naziva pravilo tri sigme. Prikaz normalne distribucije sa standardnim devijacijama je na slici [Slika 2].



Slika 2. Normalna distribucija [6]

2.5. Latin hypercube metoda uzorkovanja

Latin hypercube metoda za uzimanje uzoraka je statistička metoda za generiranje slučajnih uzoraka s vrijednostima parametara iz višedimenzionalne distribucije. Latin kvadrat je kvadratna mreža koja sadrži po samo jedan uzorak iz pojedinog reda i pojedinog stupca. Latin hypercube naziv dolazi ako taj koncept Latin kvadrata primjenimo na veći broj dimenzija. Uzorkovanje se provodi tako da ako imamo *N* varijabli, raspon svake varijable je podijeljen na *M* jednako vjerojatnih intervala odnosno segmenata. *M* točaka uzoraka je zatim raspoređeno tako da zadovoljava uvjete Latin hypercube-a. Koristi se s Monte Carlo simulacijom kako bi raspodjela velikog broja uzoraka bila šire i više jednoliko raspoređena. Prikaz Latin hypercube uzorkovanja je prikazan na slici [Slika 3]:



Slika 3. Latin hypercube uzorkovanje [8]

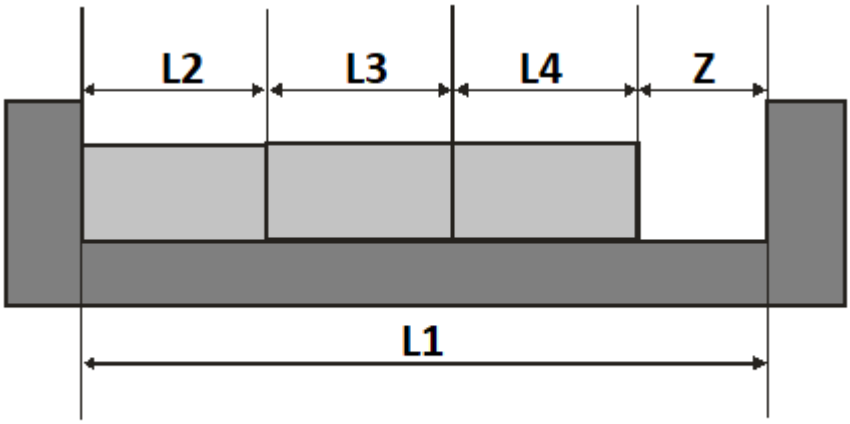
3. PRIMJENA MONTE CARLO SIMULACIJE

3.1. Statističke tolerancije

Tolerancija je dopušteno odstupanje od izmjere u proizvodnji, a jednaka je razlici najveće i najmanje dopuštene izmjere. Statističke tolerancije prvi put su obrađene u normama ASME Y14.5M-1994. i ASME Y14.5.1M-1994. U normama su u kratkim crtama naznačeni temeljni principi aritmetičkih i statističkih tolerancija uz prijedlog simbola za označavanje statističkih tolerancija na tehničkim crtežima [9]. Analiza tolerancija je proces u kojem se određuju varijacije koje su moguće između dva ili više dijelova, što se naziva slaganje tolerancija (*tolerance stackup*). Statističke tolerancije su bazirane na masovnoj proizvodnji jer „worst case" metode u kojima se uzimaju najgori slučajevi mogu biti jako skupe. Koristi se kada se treba potvrditi zračnost ili preklop između dijelova. Svaka varijacija tolerancije dijela gleda se kao slučajna, pa stoga normalna distribucija najbolje opisuje raspon tolerancije. Monte Carlo simulacija se koristi radi lakšeg izračuna ako postoji velik broj slaganja tolerancija. Također kako se statističke tolerancije koriste u masovnoj proizvodnji, velik broj uzoraka Monte Carlo simulacije se koristi kao model proizvedenih dijelova. Greška izvan zadanih granica koja se dobije simulacijom predstavlja proizvode koji se odbacuju te se iz toga može ustvrditi trošak proizvodnje. Isplati li se na tako veliku proizvodnju izvoditi finija obrada koja je skuplja da bi imali manje odbačenih proizvoda ili je korisnije ostati na istoj toleranciji s gubitkom odbačenih proizvoda. Kao parametri simulacije se uzimaju srednje vrijednosti dimenzija dijela te standardne devijacije. U praksi se najčešće za odstupanje koristi pravilo tri sigme, to jest standardna devijacija se uzima kao trećina tolerancije (tolerancija=3σ). Grafički rezultati su najčešće prikazani histogramom, na apscisi su dobivene srednje vrijednosti, a na ordinati frekvencija što predstavlja koncentraciju podataka u svakoj točki apscise.

3.1.1. Slaganje tolerancija

Sa slaganjem tolerancija se najčešće susreće kod formiranja sklopova. Dimenzije čine lanac koji je moguće algebarski zbrajati i oduzimati. Kod većih lanaca dimenzija, veće je slaganje tolerancija, te provedbom Monte Carlo simulacije se može vidjeti koje tolerancije imaju najveći utjecaj na krajnji proizvod te kako smanjiti troškove. Na slici [Slika 4] je prikaz slaganja tolerancija, jedan od primjera na koji način se tolerancije mogu slagati. Dimenzije su posložene tako da postoji vanjska mjera koja je najveća (*L1*) i od nje se oduzimaju unutranje mjere (*L2, L3, L4*) i kao rezultat se dobiva zračnost (*Z*).



Slika 4. Primjer slaganja tolerancija

3.1.2. Strujni krug

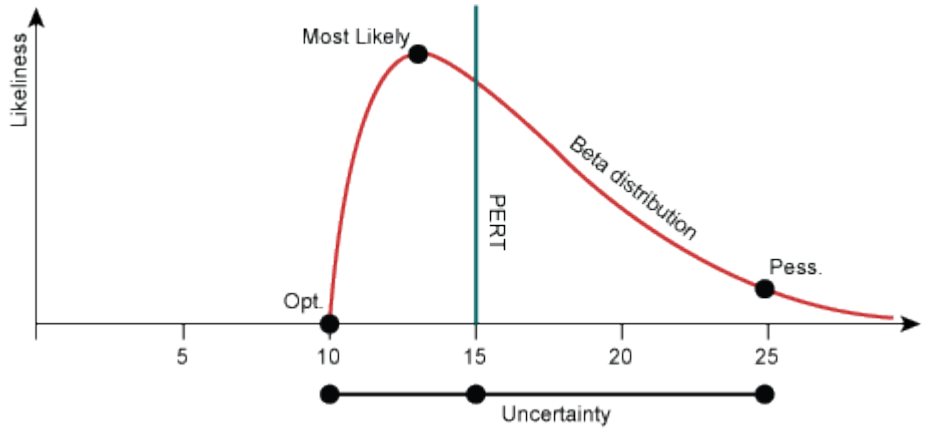
Osim slaganja tolerancija kod formiranja sklopova, mogu se uzimati u obzir i tolerancije elektroničkih komponenata kod analize električnih mreža. Sve elektroničke komponente imaju tolerancije, varijacije na njihovu nominalu vrijednost. Otpornici mogu imati 5%, 10%, izvori od 1% do 10% i tako dalje. Strujni krug mora biti izrađen da zadovoljava određene uvjete i varijacije koje unose elektroničke komponente. U primjeru s jednostavnim strujnim krugom koji ima izvor i dva otpornika spojena serijski, kao ulazne varijable se uzimaju nominalne vrijednosti izvora i otpornika te njihove tolerancije. Kao izlaz je tražena srednja vrijednost struje i njena standardna devijacija. U Monte Carlo simulaciju se unose ti parametri ulaznih varijabli i preko Ohmovog zakona se dobiva srednja vrijednost struje koja prolazi tim strujnim krugom.

3.1.3. Procjenjivanje trajanja projekta

PERT, Program Evaluation and Review Technique, je metoda u projektnom menadžmentu kojom se može predstaviti trajanje aktivnosti u danom projektu. Zadaju se tri procjene za svaku aktivnost: optimistično (a), normalno (m) i pesimistično (b) vrijeme. Očekivano vrijeme trajanja aktivnosti T_e dobije se iz formule (16):

$$Te = \frac{a + 4m + b}{6} \tag{16}$$

PERT koristi Beta distribuciju koja je statistička distribucija vjerojatnosti. Može biti simetrična ili zakrivljena, ovisno o parametrima α i β i definirana je na intervalu [0,1]. Zatim se ta metoda provede u Monte Carlo simulaciji i dobije se procijenjeno vrijeme.



Slika 5. PERT i Beta distribucija [10]

4. PYTHON I web2py MREŽNI OKVIR

4.1. Python programski jezik

Python je interpreterski, interaktivni, objektno orijentirani programski jezik [11]. Interpreterski znači da se programski kôd direktno izvršava uz pomoć interpretora. Programski kôd se piše naredba za naredbom. Objektno orijentiran je jer korisnici mogu kontrolirati objekte to jest strukture podataka, kako će program biti napisan i izvršen. Veoma je moćan, fleksibilan programski jezik i može se koristit za razvoj web aplikacija. Lakom instalacijom je moguće koristiti vanjske module (Matplotlib, NumPy, SciPy) pozivaju se naredbom *import*. Matplolib je Python knjižnica za 2D plotanje u kojoj se mogu izrađivati grafovi, histogrami sa samo par linija koda. NumPy je Python knjižnica koja podržava velike, višedimenzionalne nizove i matrice kao i kolekciju visoko naprednih matematičkih operacija za računanje s tim nizovima odnosno matricama. SciPy je također Python knjižnica koja se koristi za znanstveno i tehničko računanje. Sadrži module za optimizaciju, integraciju, interpolaciju i mnoge druge. SciPy Stats je modul koji sadrži statističke funkcije i algoritme. Jedna od osobina Python programskog jezika je da se mogu kreirati klase (*class*). Klase sadrže objekt klase koji se može pozivati kao da je funkcija. Također ima atribute koji mogu biti podatčani ili funkcijski objekti. Funkcije su skupine naredbi koje se izvršavaju po pozivu, definiraju se naredbom *def*. U Python-u je također omogućen rad s različitim vanjskim datotekama. Za rad s excel datotekama (.xlsx) dovoljno je instalirati i pozvati modul openpyxl koji je novija verzija modula za rad s excel tablicama i u njemu je omogućen direktan pristup vrijednostima koje sadrže ćelije tablice.

4.2. Općenito o web2py

Web2py je slobodni mrežni okvir za izradu web aplikacija pisan programskim jezikom Python. Baziran je na bazi podataka i sadrži sve komponente potrebne za izradu funkcionalne web aplikacije. Dizajniran je tako da koristi Model View Controller (MVC) uzorak. Razdvaja predstavljanje podataka (*the model*) od prikaza podataka (*the view*) kao i logiku i tijek rada aplikacije (*the controller*). *Model* sadrži baze podataka i sve funkcije koje sadrži su globalne, što znači da ih se može pozivati u *controller-u* i *view-u*. *Controller* i *view* su povezani na način da funkcije u *controller-u* imaju isti naziv kao *view* te se tako omogućava rad između njih. Web2py sadrži već ugrađene web forme koje korisnik može ispunjavati koristeći gumbe, *checkboxes* ili tekstualna polja. Uključuje ugrađene komponente za sve bitne funkcije kao: HTTP requests, HTTP responses, HTML/XML, JSON, SQL, DAL, jQuery za Ajax i mnoge druge.

4.3. Implementiranje Python modula mcerp u web2py

Python modul mcerp 0.11 je stohastički kalkulator za Monte Carlo metodu koji koristi Latin hypercube uzorkovanje kako bi izveo propagaciju greške (za neizvjesne analize). Koristi Python pakete NumPy i SciPy Stats te može konstruirati statističke distribucije, računati vjerojatnosti i ostalo. Paket NumPy se poziva kao *np.naredba*, a SciPy Stats kao *ss.naredba*. Modul nema mrežne mogućnosti te je stoga implementiran u web aplikaciju. Sve funkcije su unešene u novi *model* pod nazivom mcerp.py kako bi bile globalne varijable to jest mogle se pozivati u *controller-u* i *view-u*.

4.3.1. Definiranje funkcije Latin hypercube-a s Monte Carlo metodom

Funkcija *lhd* generira slučajne brojeve Latin hypercube metodom bazirane na zadanoj distribuciji. Parametri funkcije su distribucija (*dist*), broj uzoraka (*velicina*), koliko je distribucija zadano (*dims*), generiranje će se izvoditi nasumično (*form='randomized'*). Funkcija vraća izlazne varijable u obliku dvodimenzionalnog polja gdje stupci odgovaraju ulaznim varijablama, a redovi odgovaraju broju uzoraka (*out*). [Slika 6]

```
2   import numpy as np
3   import scipy.stats as ss
4   from copy import copy
5
6
7   def lhd(dist=None,velicina=None,dims=1,form='randomized',iterations=100,
8           showcorrelations=False):
9       assert dims>0,'"dims" mora biti najmanje 1 '
10      if not velicina or not dist:
11          return None
```

Slika 6. Definiranje funkcije lhd

Funkcija *_lhs* je ugnježđena funkcija *lhd*, vraća Latin hypercube matricu gdje svaki red ima različit skup slučajno uzorkovanih varijabli, koristi već zadan broj uzoraka 20, koji će se u kasnijim koracima moći promijeniti na željeni broj uzoraka. Parametar *x* je distribucija koju želimo da prate slučajno generirani brojevi. Prvo je potrebno odrediti veličinu segmenta na kojem će se uzorkovati brojevi, svi segmenti su jednake veličine. Zatim odrediti dimenziju zadane distribucije to jest broj stupaca koji će se uzorkovati. Kreiraju se matrice za izlazne varijable naredbom *np.zeros* koje se popunjavaju segment po segment sa slučajnim brojevima. Vrijednosti svakog stupca se kreću od najmanje do najveće. Slučajno generirani broj je točka kojoj se zatim pripisuje vrijednost i stavlja u matricu izlaznih varijabli. Vrijednosti svakog

stupca se kreću od najmanje do najveće. Brojevi se generiraju naredbom *np.random.random*(). Izlaz funkcije *_lhs* je izmiješana matrica izlaznih varijabli. Prikaz koda je na slici [Slika 7].

```
13      def _lhs(x, uzorci=20):
14          velicina_segmenta = 1.0/uzorci
15          stupci = x.shape[1]
16          out = np.zeros((uzorci,stupci))
17          vrijednost_tocke = np.zeros(uzorci)
18          for n in range(stupci):
19              for i in range(uzorci):
20                  segmentMin = i*velicina_segmenta
21                  tocka = segmentMin + (np.random.random()*velicina_segmenta)
22                  vrijednost_tocke[i] = (tocka*(x[1,n]-x[0,n])) + x[0,n]
23              out[:,n] = vrijednost_tocke
24          return _mix(out)
```

Slika 7. Defniranje funkcije _lhs

Ugnježđena funkcija *_mix* funkcije *lhd*, uzima podatke iz matrice izlaznih podataka i permutira ih svaki stupac posebno kako ne bi bili poredani od najmanjeg do najvećeg. Pomoću *data.shape*[0] se dobiva broj redova, koji se sprema u *n*. Novi poredak podataka to jest njegovi retci se permutiraju naredbom *np.radnom.permutation()*. Ako imamo matricu [3x4] njen *data_poredak* će iznositi [0, 1, 2], a permutacijom se može dobiti na primjer [1, 2, 0]. Naredbom *np.unique* se uzimaju samo po jedan to jest jedinstven od istih brojeva od prvotnog i novog poretka podataka koji su složeni horizonatlno naredbom *np.hstack*, složeni su u niz. U *poredak* se spremaju u niz stari i novi poredak podataka. U primjeru s navedenom matricom bi to bilo [0 1 2 1 2 0]. Pomoću kriške (eng. *slicing*) odvaja se stari od novog poretka, *poredak*[:n] uzima sve vrijednosti do *n* (primjer s matricom, uzima prve 3 vrijednosti). Dok *poredak*[-n:] uzima zadnjih *n* vrijednosti. Zatim se naredbom *np.argsort* sortira niz to jest nova matrica redoslijeda varijabli po uzoru na staru, varijabla *i* uzima stupce (*data.shape*[1]) i sprema ih po novom poretku. U krajnju *data* se spremaju sve permutirane vrijednosti. [Slika 8]

```
31      def _mix(data, dim='cols'):
32          data = np.atleast_2d(data)
33          n = data.shape[0]
34
35          if dim=='rows':
36              data = data.T
37
38          data_poredak = range(n)
39          for i in range(data.shape[1]):
40              novi_data_poredak = np.random.permutation(data_poredak)
41              vals, poredak = np.unique(np.hstack((data_poredak, novi_data_poredak)), return_inverse=True)
42              stari_poredak = poredak[:n]
43              novi_poredak = poredak[-n:]
44              tmp = data[np.argsort(stari_poredak), i][novi_poredak]
45              data[:, i] = tmp[:]
46
47          if dim=='rows':
48              data = data.T
49
50          return data
```

Slika 8. Definiranje funkcije _mix

Nasumično generiranje brojeva uzima uniformno distribuirane brojeve iz funkcije *_lhs* i njih transformira u brojeve distribuirane po željenoj razdiobi, metodom inverzne transformacije, naredba *ppf* je inverz kumulativne razdiobe. Ako je ulaz jedna distribucija, *dist_data* pravi prazan niz (matricu) naredbom *np.empty_like*, po uzoru na matricu uniformnih brojeva (*unif_data*), te transformira njene stupce (*nvars*) u vrijednosti željene distribucije preko naredbe *dist.ppf*. Prikaz koda je na slici [Slika 9]

```
47      if form is 'randomized':
48          if hasattr(dist,'__getitem__'): # ulaz su vise distribucija
49              nvars = len(dist)
50              x = np.vstack((np.zeros(nvars),np.ones(nvars)))
51              unif_data = _lhs(x,uzorci=velicina)
52              dist_data = np.empty_like(unif_data)
53              for i,d in enumerate(dist):
54                  dist_data[:,i] = d.ppf(unif_data[:,i])
55
56          else: # ulaz je jedna distribucija
57              nvars = dims
58              x = np.vstack((np.zeros(nvars),np.ones(nvars)))
59              unif_data = _lhs(x,uzorci=velicina)
60              dist_data = np.empty_like(unif_data)
61              for i in range(nvars):
62                  dist_data[:,i] = dist.ppf(unif_data[:,i])
```

Slika 9. Slučajno generiranje brojeva

4.3.2. Definiranje funkcija distribucije

Klasa *NeizvjesnaFunkcija* inicijalno uzima sve vrijednosti svih generiranih varijabli i sprema ih pod nazivom *_mcpts*. Također definira srednju vrijednost naredbom *np.mean*. Funkcija *var* vraća varijancu funckije, a funkcija *std* vraća standardnu devijaciju funkcije što je korijen varijacije. Varijabla *npts* je ukupan broj uzoraka za koji će se izvršiti simulacija, postavljen je na 100 000. Prikaz linija koda je na slici [Slika 10]. Nije potrebno postaviti veći broj simulacija jer poslije tog broja uzoraka rezultati se skoro ni ne mijenjaju, a veći broj uzoraka bi usporio rad aplikacije.

```
154  npts = 100000
155
156  class NeizvjesnaFunkcija(object):
157      def __init__(self, mcpts):
158          self._mcpts = np.atleast_1d(mcpts).flatten()
159          self.tag = None
160
161      @property
162      def mean(self):
163          mn = np.mean(self._mcpts)
164          return mn
165
166      @property
167      def var(self):
168          mn = self.mean
169          vr = np.mean((self._mcpts - mn)**2)
170          return vr
171
172      @property
173      def std(self):
174          return self.var**0.5
```

Slika 10. Definiranje klase NeizvjesnaFunkcija

Klasa *NeizvjesnaVarijabla* inicijalno generira vrijednosti iz funkcije *lhd* s naredbom *rv* koja vraća slučajne varijable distribucije, bilo neprekidne ili diskretne ovisno o željenoj distribuciji, a poziva se iz SciPy.Stats modula. Za lakšu upotrebu poziva se naredbom *uv* (engleski *Uncertain Variable*). Prikaz programskog koda na slici [Slika 11].

```
399  class NeizvjesnaVarijabla(NeizvjesnaFunkcija):
400      def __init__(self, rv, tag=None):
401          assert hasattr(rv, 'dist'), 'Ulaz mora biti distribucija iz ' + \
402              'the scipy.stats module.'
403          self.rv = rv
404          # generiraj latin-hypercube točke
405          self._mcpts = lhd(dist=self.rv, velicina=npts).flatten()
406          self.tag = tag

441  uv = NeizvjesnaVarijabla
```

Slika 11. Definiranje klase NeizvjesnaVarijabla

4.3.2.1. Normalna distribucija

Funkciji *Normal* zadaju se parametri srednja vrijednost (*mu*) i standardna devijacija (*sigma*) i funkcija vraća normalnu distribuciju s tim parametrima pomoću naredbe *ss.norm* koja pripada klasi *NeizvjesnaVarijabla* (*uv*). [Slika 12]

```
439  def Normal(mu, sigma, tag=None):
440
441      assert sigma>0, 'Normal "sigma" mora biti veca od 0'
442      return uv(ss.norm(loc=mu, scale=sigma), tag=tag)
443
444  N = Normal  # za lakšu upotrebu
```

Slika 12. Definiranje funkcije Normal

4.3.2.2. PERT metoda

Funkcija *PERT* uzima kao parametre donju granicu (*low*), vršnu vrijednost (*peak*) i gornju granicu (*high*). Vraća *Beta* distribuciju s izračunatim parametrima, kao parametrima za *alfa* i *beta* za *Beta* distribuciju. [Slika 13]

```
447  def Beta(alpha, beta, low=0, high=1, tag=None):
448
449      assert alpha>0 and beta>0, 'Beta "alpha" and "beta" parametri moraju biti veci od nula'
450      assert low<high, 'Beta "low" mora biti manji od "high"'
451      return uv(ss.beta(alpha, beta, loc=low, scale=high-low), tag=tag)

453  def PERT(low, peak, high, g=4.0, tag=None):
454
455      a, b, c = [float(x) for x in [low, peak, high]]
456      assert a<=b<=c, 'PERT "peak" mora biti veci od "low" i manji od "high"'
457      assert g>=0, 'PERT "g" mora biti nenegativan'
458      mu = (a + g*b + c)/(g + 2)
459      if mu==b:
460          a1 = a2 = 3.0
461      else:
462          a1 = ((mu - a)*(2*b - a - c))/((b - mu)*(c - a))
463          a2 = a1*(c - mu)/(mu - a)
464
465      return Beta(a1, a2, a, c, tag)
```

Slika 13. Definiranje PERT metode

4.3.3. Definiranje vjerojatnosti

Kako bi se odredio udio distribucije koji je ispod ili iznad neke točke u distribuciji, dovoljno je koristit standardne znakove usporedbe < ili >. Funkcija *__le__* uspoređuje koliki je udio distribucije ispod neke točke (*val*), provjerava pripada li *val Neizvjesnoj Funkciji*. Vraća duljinu tog udjela (broj uzoraka na tom dijelu) i dijeli s ukupnim brojem uzoraka i to pomnoženo sa 100 daje postotak udjela koji je ispod te točke. Funkcija *__ge__* uspoređuje koliki je udio

distribucije iznad neke točke. Koristi se standardizirana distribucija koja prima vrijednosti od 0 do 1. Obje funkcije pripadaju klasi *NeizvjesnaFunkcija*. Prikaz dijela programskog koda je na slici [Slika 14].

```
366     def __le__(self, val):
367         if isinstance(val, NeizvjesnaFunkcija):
368             return self<val
369         else:
370             return len(self._mcpts[self._mcpts<=val])/float(npts)
371 
387 
388     def __ge__(self, val):
389         if isinstance(val, NeizvjesnaFunkcija):
390             return self>val
391         else:
392             return 1 - (self<val)
```

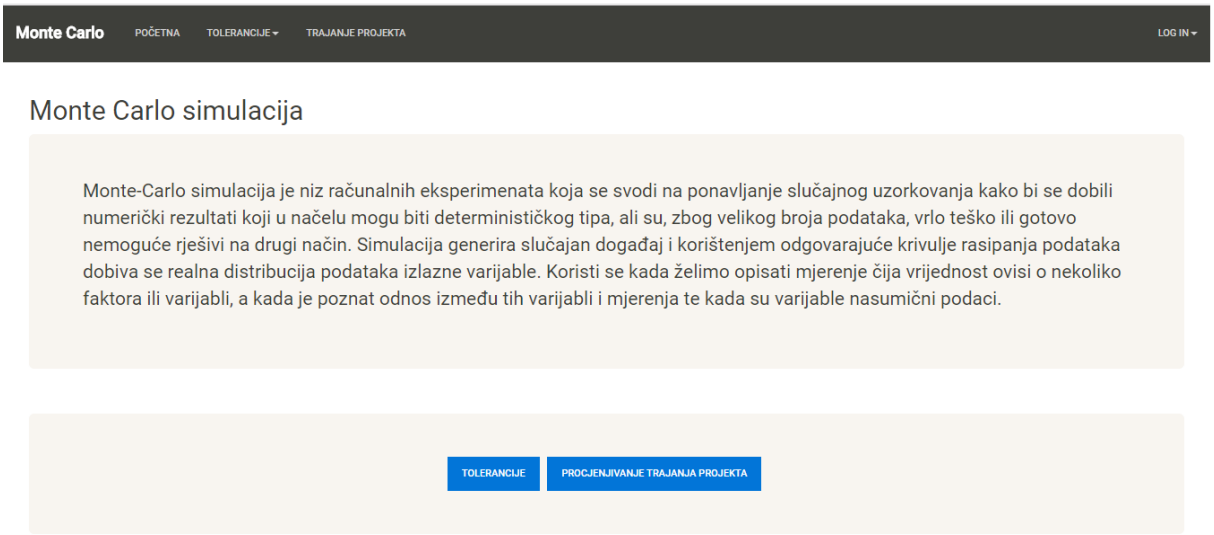
Slika 14. Definiranje funkcija __le__ i __ge__

5. ON-LINE SIMULACIJA

U web2py mrežnom okviru je izrađena web aplikacija za on-line Monte Carlo simulaciju. Prvo je prikazan izgled stranice i način korištenja, a kasnije u poglavlju je prikaz *modela*, *viewa* i *controllera* aplikacije. Otvaranjem aplikacije otvara se Početna stranica na kojoj je moguć odabir primjera vezanih za Monte Carlo simulaciju. Pritiskom na gumb *Tolerancije* ili *Trajanje projekta* otvaraju se primjeri za provedbu simulacije. Otvaranjem bilo od kojeg primjera tolerancija moguće je unesti podatke odabirom excel datoteke u kojoj su zapisani podaci s kojim se računa. Simulacija vraća rezultate simulacije kao i histogram rezultata. Primjer trajanje projekta omogućuje ručni unos vremena aktivnosti koji se procjenjuje.

5.1. Početna stranica

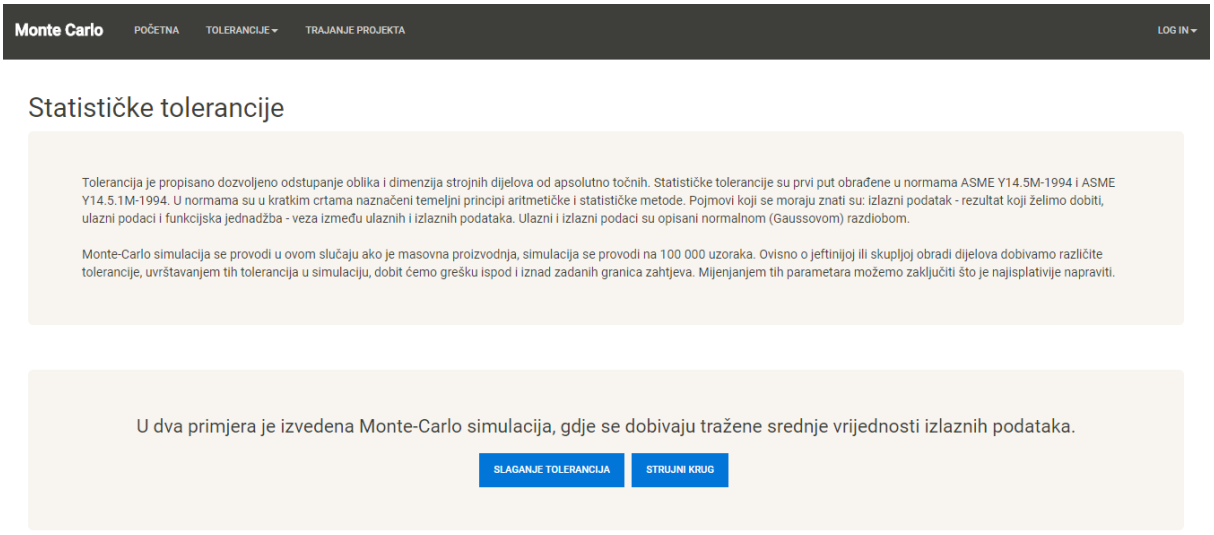
Na početnoj stranici je općenit opis Monte Carlo metode i na dnu stranice se nalaze gumbi za odabir primjera, *tolerancije* ili *trajanje projekta*. Na vrhu stranice se nalazi izbornik na kojem se također može odabrati ili povratak na početnu stranicu ili tolerancije koje imaju padajući izbornik za slaganje tolerancija i strujni krug, ili procjenjivanje trajanja projekta. [Slika 15]



Slika 15. Početna stranica

5.2. Stranica Tolerancije

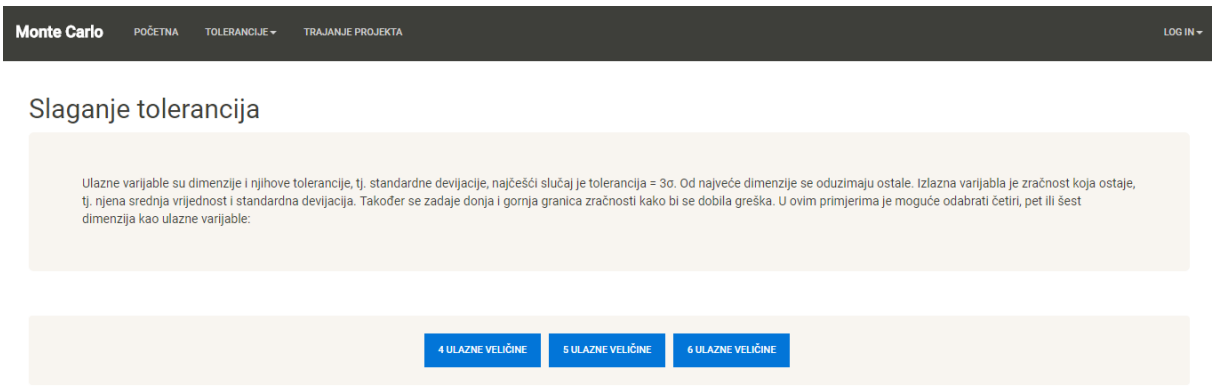
Pritiskom na gumb *tolerancije* se otvara stranica s primjerima za tolerancije. Na stranici je općenit opis toga što su tolerancije, a na dnu je moguć odabir vrste primjera, *slaganje tolerancija* ili *strujni krug*. [Slika 16]



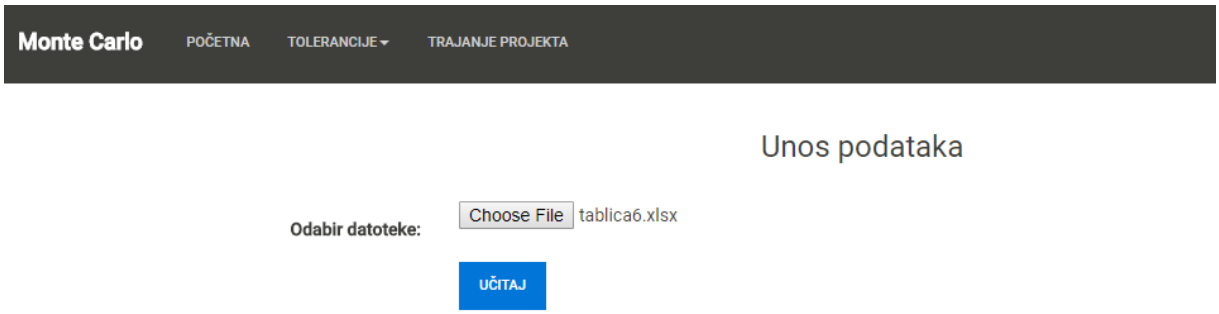
Slika 16. Stranica Statističke tolerancije

5.2.1. Simulacija slaganja tolerancija

Ako se odabere slaganje tolerancija, otvara se nova stranica na kojoj je općenit opis što je slaganje toleracija. Moguć je odabir simulacije s četiri, pet ili šest dimenzija na dnu stranice [Slika 17] kojima se računa zračnost. Pritiskom na jedan od tih gumbova se otvara stranica na kojoj se odabire excel datoteka s unesenim podacima [Slika 18].



Slika 17. Stranica slaganje tolerancija



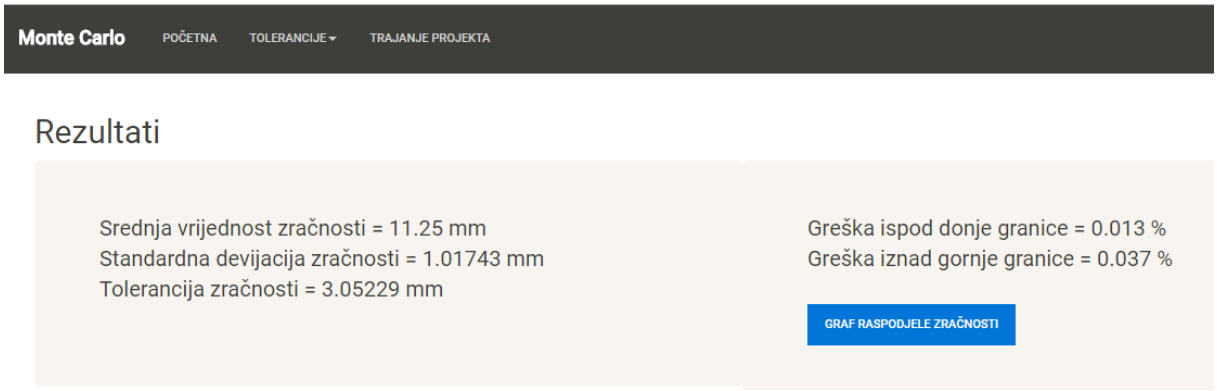
Slika 18. Unos podataka

Podaci u tablici su uneseni na način da su u prvom stupcu srednje vrijednosti dimenzija, prva je najveća, vanjska dimenzija i zatim ostale dimenzije. U drugom stupcu su standardne devijacije tim dimenzija koje iznose trećinu tolerancije. I u trećem stupcu je prvo unesena donja granica pa gornja granica zračnosti. [Slika 19].

	A	B	C	D
1	93	0.5	7.5	
2	23	0.33333	14.8	
3	13.5	0.5		
4	20.25	0.25		
5	17	0.5		
6	8	0.33333		
7				
8				
9				

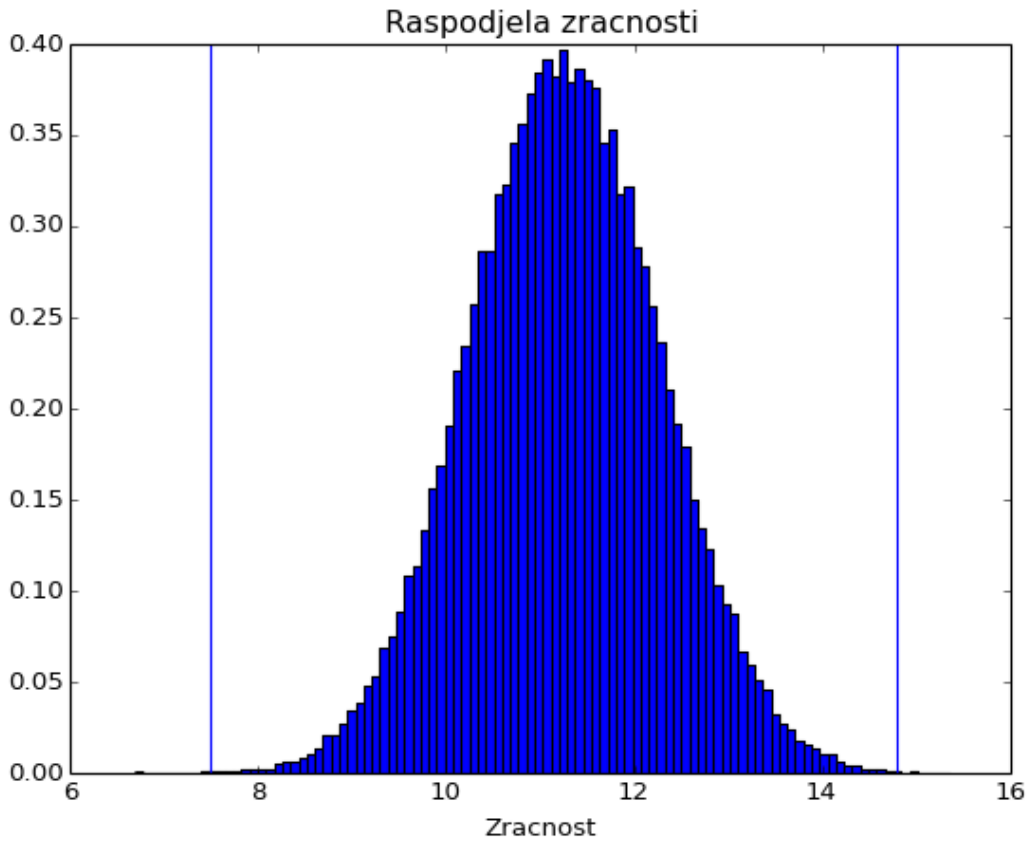
Slika 19. Excel tablica

Pritiskom na *učitaj* učitava se odabrana datoteka te se izračuna srednja vrijednost, standardna devijacija i tolerancija zračnosti. Također se prikazuje greška u postotcima ispod i iznad zadanih granica. Dobivena greška od 0.037% znači da će na postavljenih 100 000 uzoraka, 37 biti izvan gornje granice to jest biti nezadovoljavajući. [Slika 20]



Slika 20. Prikaz rezultata slaganja tolerancija

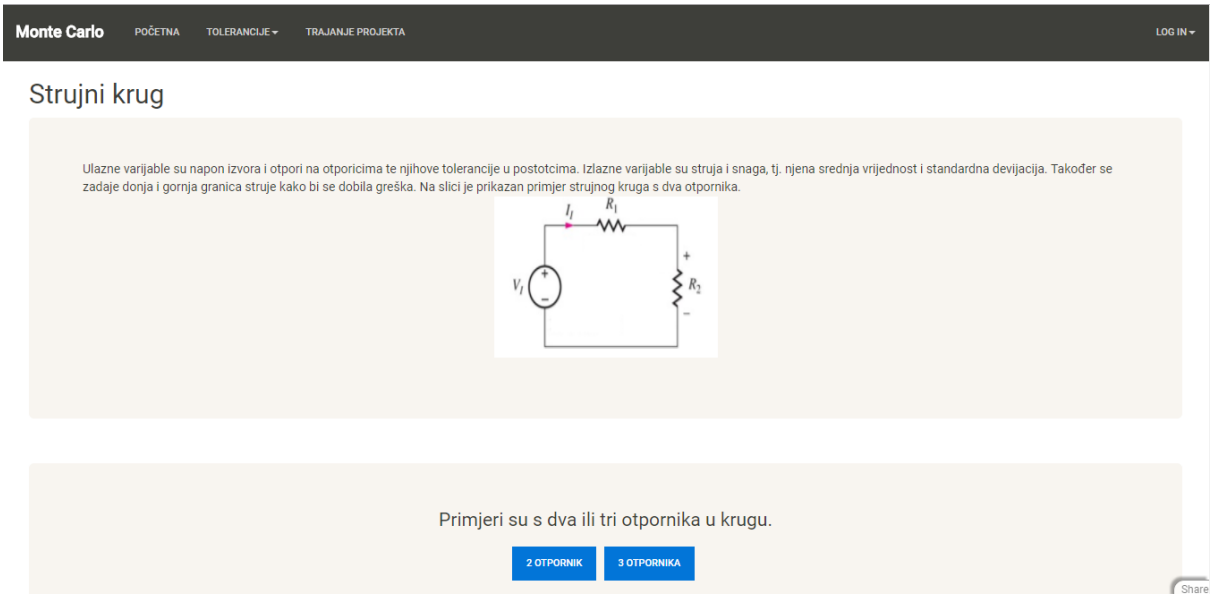
Moguće je prikazati graf raspodjele zračnosti odnosno histogram, koji na x osi ima srednju vrijednost zračnosti, a na y osi frekvenciju to jest koncentraciju podataka. Graf se otvara kao slika pa ju je moguće usnimiti direktno na računalo [Slika 21].



Slika 21. Histogram raspodjele zračnosti

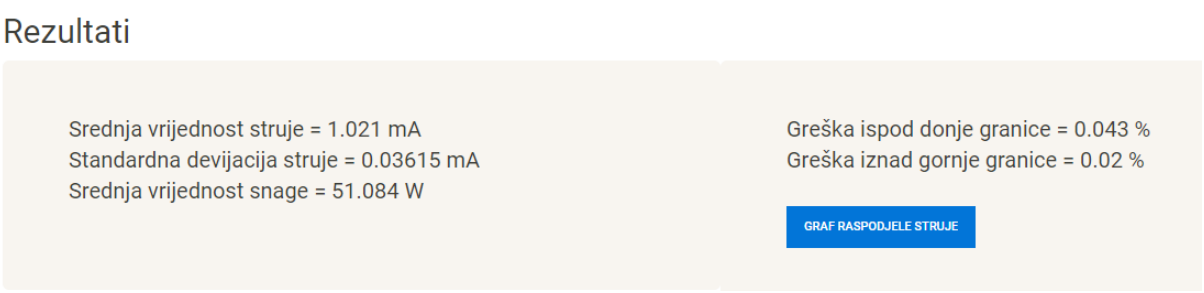
5.2.2. Simulacija strujnog kruga

Ako je pritisnut gumb *strujni krug* otvara se stranica na kojoj je opis što su ulazne, a što izlazne varijable. Moguć je odabir od dva primjera strujnog kruga na dnu stranice pomoću gumba. Strujni krug koji ima izvor i serijski spojena dva ili tri otpornika [Slika 22]. Unos podataka se odvija na isti način kao u prošlom primjeru [Slika 18]. U tablici za unos podataka su u prvom stupcu nominalne vrijednosti izvora i otpornika. U drugom stupcu su njihove tolerancije u postotcima. I u trećem stupcu je na prvom mjestu donja granica struje, a na drugom gornja granica. Podaci se unose u voltima i kiloohmima.

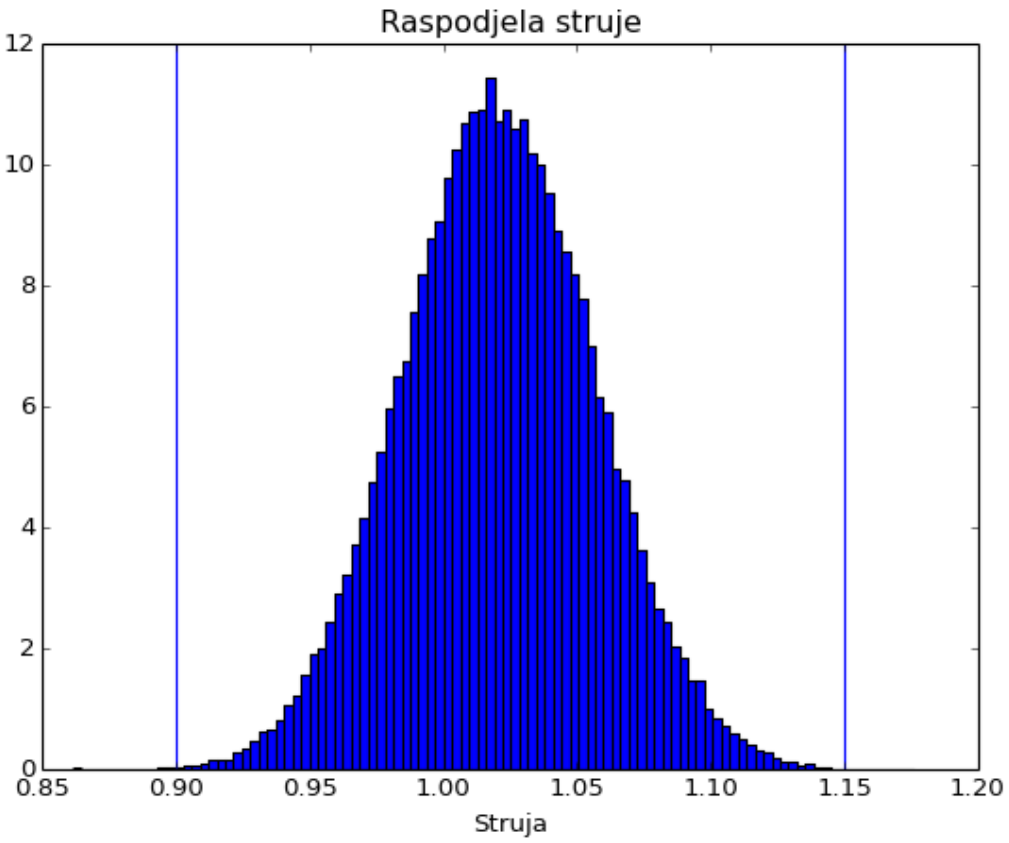


Slika 22. Stranica Strujni krug

Rezultati prikazuju srednju vrijednost struje u miliamperima, njenu standardnu devijaciju, srednju vrijednost snage i grešku izvan zadanih granica [Slika 23]. Također je moguć prikaz histograma raspodjele struje. [Slika 24].



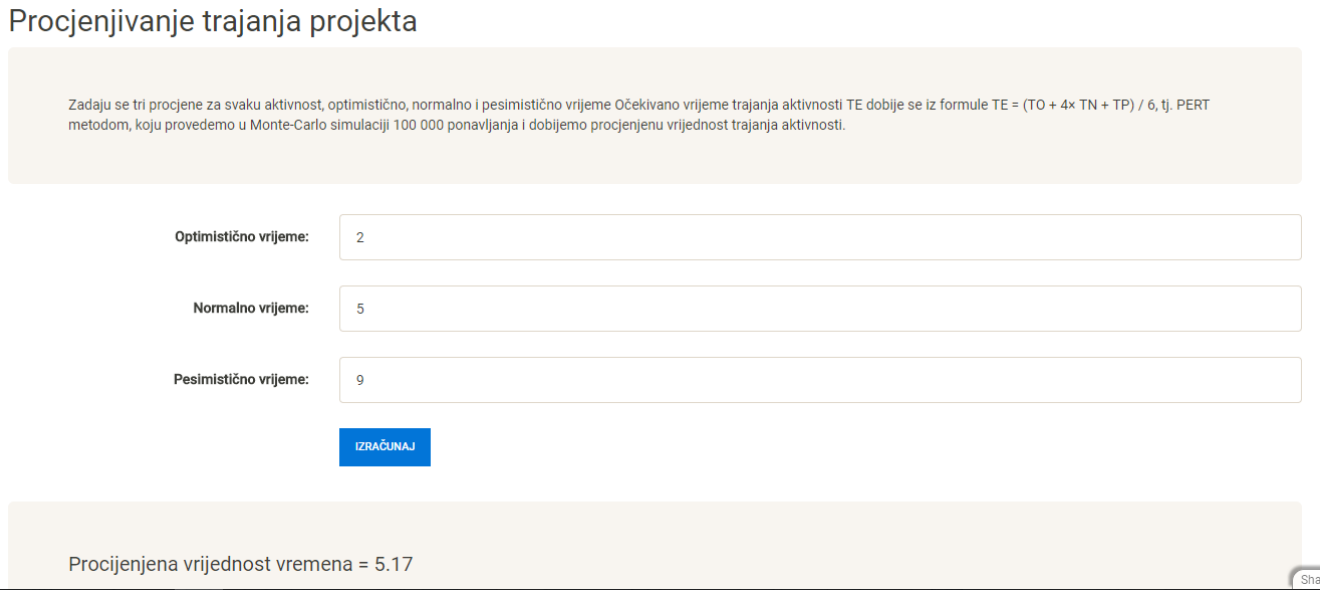
Slika 23. Prikaz rezultata struje i snage



Slika 24. Prikaz histograma struje

5.3. Stranica trajanje projekta

Kao netehnički primjer izrađena je simulacija za izračun trajanja nekog projekta to jest aktivnosti. Unosi se optimistično, normalno i pesimistično vrijeme aktivnosti te se dobije procjena trajanja te aktivnosti. [Slika 25]



Slika 25. Procjenjivanje trajanja projekta

5.4. Model web aplikacije

U *modelu* web aplikacije su kreirane baze podataka. Naredbom *db.define_table* se stvara tablica naziva „*tolerancije_1*“ koja ima polje za unos datoteke [Slika 26]. Sve unešene datoteke se spremaju u bazu podataka koja ima naziv „*tolerancije_1*“. Polje *unos* tipa *upload* znači da će se unositi vanjska datoteka s podacima. Na isti način su stvorene baze podataka za ostale primjere s tolerancijama.

```
134  db.define_table('tolerancije_1',
135      Field('unos','upload',label='Datoteka'))
```

Slika 26. Kreiranje baze podataka

Također je kreirana baza podataka pod nazivom „*projekt*“, koja ima 4 polja koja su tipa *double* odnosno decimalni brojevi. *Readable* i *writable* znači da se to polje ne može unositi, ali je u njega moguće spremati podatke [Slika 27].

```
157 db.define_table('projekt',
158     Field('Topt', 'double',label='Optimistično vrijeme:'),
159     Field('Tnor', 'double',label='Normalno vrijeme:'),
160     Field('Tpes', 'double',label='Pesimistično vrijeme:'),
161     Field('srednja', 'double', readable=False, writable=False))
```

Slika 27. Baza podataka „projekt“

5.5. Controller web aplikacije

U *controlleru* se određuje tijek rada aplikacije. Pozivaju se sve kreirane funkcije u *modelu* kao i baze podataka. Potrebno je kreirati funkciju za svaki poseban *view* pod istim nazivom kao funkcija, u koji će se pozivati ono što je definirano pod tom funkcijom. SQLFORM kreira izgled forme kojom se unose podaci u bazu podataka *tolerancije_1* (*db.tolerancije_1*). Ako je forma prihvaćena, to jest podaci su uneseni, automatski nas aplikacija odvodi na prikaz rezultata [Slika 28]. Funkcija *def tolerancije_1* je povezena s stranicom za unos podataka [Slika 18].

```
25 def tolerancije_1():
26     form = SQLFORM(db.tolerancije_1, submit_button='Učitaj')
27     if form.process().accepted:
28         session.flash = 'Podaci uneseni!'
29         redirect(URL('rezultat1'))
30     return locals()
```

Slika 28. Funkcija tolerancije_1

Za automatski prikaz rezultat unesenih podataka potrebno je prihvatit zadnju unesenu datoteku, to izvodi naredba *max.id*. [Slika 29] Za prihvaćanje excel datoteke je naredba *load_workbook*. Varijabla *first_sheet* sprema naziv prvog lista u datoteci i zatim ga poziva s *wb.get_sheet_by_name*(). Moguće je direktno uzimati vrijednosti ćelija, npr. *worksheet['A1'].value* uzima vrijednost ćelije A1 prvog lista odabrane datoteke. Pozivom funkcije *N* s parametrima koji su u ćeliji A1 i B1 (u stupcu A su srednje vrijednosti, a stupcu B standardne devijacije), se generira normalna distribucija i sprema u *L1*. Za primjer u kojem se računa zračnost kod slaganja tolerancija, varijabla *Z* predstavlja zračnost koja se dobiva preko *Z=L1-L2-L3-L4*, ovisno s koliko dimenzija se računa. Pozivom *Z.mean* i *Z.std* se dobiva srednja vrijednost i standardna devijacija raspodjele zračnosti. Varijable *ispod* i *iznad* spremaju udio raspodjele koji je ispod i izvan zadanih granica. Funkcija *def rezultat1* je povezana s prikazom rezultata kao što je na slici [Slika 20]

```
def rezultat1():
    max_id = db(db.tolerancije_1).select(db.tolerancije_1.id.max()).first()[db.tolerancije_1.id.max()]
    for redak in db(db.tolerancije_1.id==max_id).select():
        unos=redak.unos
        wb = load_workbook(os.path.join(request.folder,'uploads',unos))
        first_sheet = wb.get_sheet_names()[0]
        worksheet = wb.get_sheet_by_name(first_sheet)
        L1=N(worksheet['A1'].value,worksheet['B1'].value)
        L2=N(worksheet['A2'].value,worksheet['B2'].value)
        L3=N(worksheet['A3'].value,worksheet['B3'].value)
        L4=N(worksheet['A4'].value,worksheet['B4'].value)
        Z=L1-L2-L3-L4
        srednja=Z.mean
        devijacija=Z.std
        ispod=Z<worksheet['C1'].value
        iznad=Z>worksheet['C2'].value
    return locals()
```

Slika 29. Funkcija rezultat1

Za crtanje grafa odnosno histograma je korišten modul *matplotlib*, ali je također u *controlleru* definirana funkcija za izradu histograma s unešenim podacima. Prihvaćanje podataka je isto kao i za izračun rezultata, osim prihvaćanja svih vrijednosti zračnosti, varijable *Z*, naredbom *_mcpts*. Kako bi se mogao kreirati graf potrebno je sortirati te vrijednosti. Pozivom *myplot*, *lsl* i *usl* izrađuju dvije vertikalne linije u vrijednostima donje i gornje granice. *Mode='hist'* znači da se iscrtava histogram [Slika 30]. Za lakše spremanje grafa kao slike *response.headers['Content-Type']='image/png'* otvara graf kao sliku png formata.

```
def graf():
    response.headers['Content-Type']='image/png'
    max_id = db(db.tolerancije_1).select(db.tolerancije_1.id.max()).first()[db.tolerancije_1.id.max()]
    for redak in db(db.tolerancije_1.id==max_id).select():
        unos=redak.unos
        wb = load_workbook(os.path.join(request.folder,'uploads',unos))
        first_sheet = wb.get_sheet_names()[0]
        worksheet = wb.get_sheet_by_name(first_sheet)
        L1=N(worksheet['A1'].value,worksheet['B1'].value)
        L2=N(worksheet['A2'].value,worksheet['B2'].value)
        L3=N(worksheet['A3'].value,worksheet['B3'].value)
        L4=N(worksheet['A4'].value,worksheet['B4'].value)
        Z=L1-L2-L3-L4
        donja=worksheet['C1'].value
        gornja=worksheet['C2'].value
        y=sorted(Z._mcpts)
        return myplot(lsl=[donja],usl=[gornja],data={'data':[(0,y)]},mode='hist',title='Raspodjela zračnosti',xlab='Zračnost')
```

Slika 30. Funkcija graf

Kod računanja vrijednosti struje princip rada je isti, razlika je u povezivanju izlaznih varijabli s ulaznim i pretvaranje postotka tolerancije u standardnu devijaciju. Prikaz *controllera* je na slici [Slika 31], a povezan je s prikazom rezultata na [Slika 23].

```
168 def rezultat4():
169     max_id = db(db.struja_1).select(db.struja_1.id.max()).first()[db.struja_1.id.max()]
170     for redak in db(db.struja_1.id==max_id).select():
171         unos=redak.unos
172         wb = load_workbook(os.path.join(request.folder,'uploads',unos))
173         first_sheet = wb.get_sheet_names()[0]
174         worksheet = wb.get_sheet_by_name(first_sheet)
175         sigma1=(worksheet['A1'].value*worksheet['B1'].value/100.0000)/3.0000
176         V1=N(worksheet['A1'].value,sigma1)
177         sigma2=(worksheet['A2'].value*worksheet['B2'].value/100.0000)/3.0000
178         R1=N(worksheet['A2'].value,sigma2)
179         sigma3=(worksheet['A3'].value*worksheet['B3'].value/100.0000)/3.0000
180         R2=N(worksheet['A3'].value,sigma3)
181         I1=V1/(R1+R2)
182         P1=V1*I1
183         Isrednja=I1.mean
184         Idevijacija=I1.std
185         Psrednja=P1.mean
186         Pdevijacija=P1.std
187         ispod=I1<worksheet['C1'].value
188         iznad=I1>worksheet['C2'].value
189     return locals()
```

Slika 31. Funkcija rezultat za iznose struje

Ako se želi nešto računati s unesenim podacima u bazi podataka potrebna je naredba *onvalidation*. Kako bi se dva polja mogla zbrajati, dijeliti, oduzimati i ostalo, potrebno je prvo definirati zasebnu funkciju prije nego je forma procesirana odnosno prihvaćena [Slika 32]. Inače ako bi bio korišten znak + u svrhu zbrajanja, nakon što je forma procesirana, npr. 1+1=2, dobili bi 11, a ne traženih 2. Pozivom *PERT* simulira se metoda s unesenim parametrima za vrijeme.

```
257 def processing_projekt(form):
258     Topt=form.vars.Topt
259     Tnor=form.vars.Tnor
260     Tpes=form.vars.Tpes
261     Tpro=PERT(Topt,Tnor,Tpes)
262     srednja=Tpro.mean
263     form.vars.srednja=round(srednja,2)
264 
265 def projekt():
266     form = SQLFORM(db.projekt, submit_button='Izračunaj')
267     if form.process(onvalidation=processing_projekt).accepted:
268         session.flash = 'Podaci uneseni!'
269     return dict(form=form)
```

Slika 32. Naredba onvalidation

5.6. View web aplikacije

U *viewu* je Python kôd pisan u zagradama {{}}, a ostalo je pisano HTML jezikom. Umjesto naredbe *print* dovoljno je staviti samo znak jednakosti. Moguće je pozvati već postojeći izgled stranice s *extend 'layout.html'*. Svi rezultati su zaokruženi na pet decimala (*round*), ako je *view* istog naziva *rezultat1* kao funkcija u *controlleru* onda će prikazivati samo rezultate tog primjera [Slika 33]. Ovdje je *rezultat1* primjer s 4 dimenzije, postoje još *rezultat2* s 5 dimenzija i *rezultat4* s 6 dimenzija. Svi su napravljeni na isti način. Primjeri sa strujnim krugom su rađeni po istom principu samo što se još prikazuje i srednja vrijednost snage.

```
{{extend 'layout.html'}}
<h2>Rezultati</h2>
<div class='jumbotron' style='width: 50%; height: 50%; float:left;font-size:20px;'>
Srednja vrijednost zračnosti = {{=round(srednja,5)}} mm
<br>
Standardna devijacija zračnosti = {{=round(devijacija,5)}} mm
<br>
Tolerancija zračnosti = {{=round(devijacija*3,5)}} mm
<br>
      
</div>
<div class ='jumbotron' style='width: 50%; height: 50%; float:right;font-size:20px'>
Greška ispod donje granice = {{=ispod*100}} %
<br>
Greška iznad gornje granice = {{=iznad*100}} %
```

Slika 33. Prikaz view-a

6. ZAKLJUČAK

Monte Carlo simulacija je koristan alat kada je potrebno izvoditi operacije s velikom količinom podataka i kod neizvjesnih problema. Na bazi slučajnosti i vjerojatnosti, dobiju se rezultati koji mogu biti deterministčkog tipa. Moguće ju je povezati s mnogo drugih matematičkih metoda, bilo metoda za uzorkovanje ili raspodjelu podataka. Web aplikacija za on-line simulaciju omogućuje korisniku jednostavno korištenje Monte Carlo simulacije. Jednostavnim unosom datoteke s već ispisanim podacima dobivaju se rezultati i histogram na već stvorenim primjerima. Prikazana je jedna od najčešće primjene Monte Carlo simulacije u strojarstvu to jest u analizi statističkih tolerancija. Kao netehnički primjer su procjene trajanja aktivnosti ili sveukupnog projekta koje se baziraju na vjerojatnostima vremena danima PERT metodom.

LITERATURA

[1] Weisstein, E. W.: Monte Carlo method, From MathWorld-A Wolfram Web Resource

[2] Sobol, I. M.: A Primer for the Monte Carlo method, CRC Press, 1994.

[3] Singer, S.: Slučajne varijable, skripta, Sveučilište u Zagrebu, FSB

[4] Du, X.: Monte Carlo simulation, Missouti University of Science and Technology, 2011.

[5] https://am207.github.io/2017/wiki/inversetransform.html

[6] http://www.muelaner.com/wp-content/uploads/2013/07/Standard_deviation_diagram.png

[7] Iman, R.L., Davenport J. M., Ziegler, D.K.: Latin Hypercube Sampling (Program User's Guide), 1980.

[8] http://www.epixanalytics.com/modelassist/AtRisk/Model_Assist.htm#Montecarlo/Latin_Hypercube_sampling.htm

[9] Mudronja V., Baršić G., Katić M., Šimunović V.: Indeksi sposobnosta procesa i statističke tolerancije, 14. Hrvatska konferencija o kvaliteti, 2014.

[10] Sherman P. J.: Better project management through beta distribution, Six Sigma, 2016.

[11] Essert, M.: Digitalni udžbenik Python, Sveučilište Josipa Jurja Strossmayera Osijek, Odjel za matematiku, 2007.

[12] Di Pierro, M.: Web2py Book, www.web2py.com/book, 2017.

[13] http://pythonhosted.org/mcerp/

PRILOZI

I. CD-R disc