UDŽBENIK ELEKTROTEHNIČKOG FAKULTETA U BEOGRADU

Milan Bjelica

MODELIRANJE I SIMULACIJA U TELEKOMUNIKACIJAMA

Beograd, 2013.

dr Milan Bjelica,
Elektrotehnički fakultet Univerziteta u Beogradu
e–mail: milan@etf.rs

MODELIRANJE I SIMULACIJA U TELEKOMUNIKACIJAMA
elektronski udžbenik

Recenzenti:

prof. dr Predrag Pejović,
prof. dr Aleksandra Smiljanić

Nastavno-naučno veće Elektrotehničkog fakulteta odobrilo je objavljivanje ovoga udžbenika odlukom broj 26 od 30.1.2013. godine.

Izdavač:

Elektrotehnički fakultet
Univerziteta u Beogradu

ISBN: 978-86-7225-053-4

Tekst ove knjige složen je u programskom paketu LaTeX 2_ε.

Sadržaj

Poglavlje 1

Uvod u modeliranje i simulaciju

Kada bi trebalo sažeti ovaj kurs u nekoliko rečenica, kazali bismo da se u njemu izučava korišćenje računara za *oponašanje* pojava i procesa iz stvarnog sveta. Prvi korak ka tome cilju je usvajanje pretpostavki koje će nam omogućiti matematički opis posmatrane pojave. U najvećem broju slučajeva, verodostojni opisi će rezultirati matematičkim relacijama koje se mogu rešiti jedino numerički, pa ćemo ovako dobijene rezultate koristiti za *procenu* stvarnih karakteristika pojave koju posmatramo.

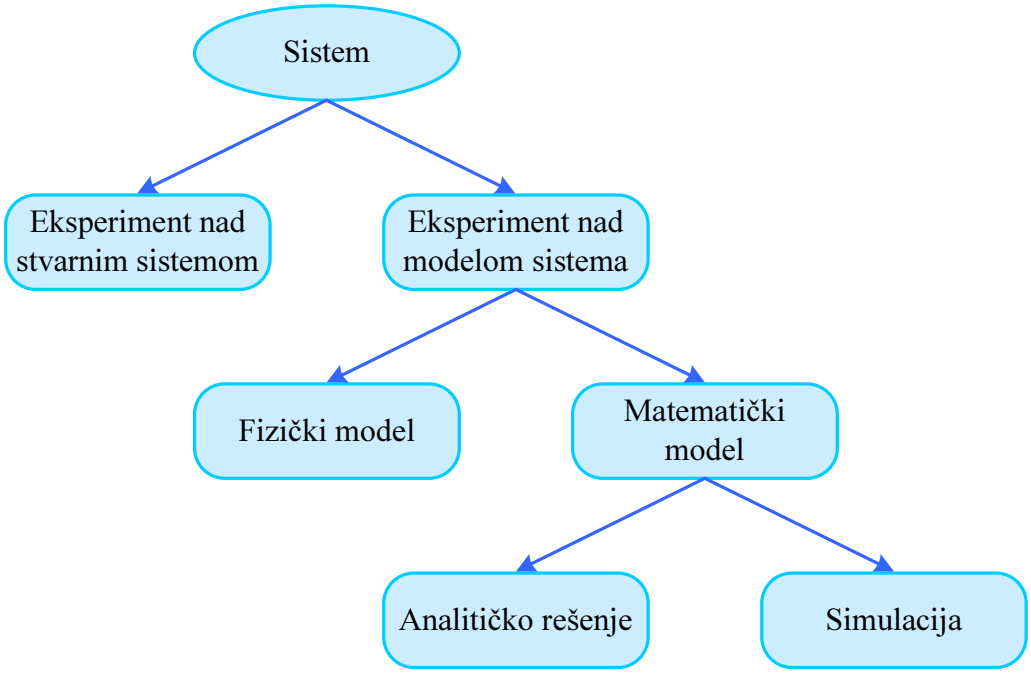
Sada ćemo detaljnije objasniti najvažnije pojmove na kojima se zasnivaju teorijske postavke simulacije.

Sistem je skup entiteta koji deluju zasebno i uzajamno ka postizanju nekog logičkog cilja. Ono što je sa stanovišta jednog cilja kompletan sistem, sa stanovišta nekog drugog može biti samo entitet u sistemu višega reda. Ako se, na primer, posmatra opsluživanje paketa na ulaznom portu jednog rutera, pridruženi red čekanja se može smatrati sistemom; s druge strane, ako se posmatra rad rutera kao celine, ovaj red čekanja predstavlja jedan njegov gradivni elemenat, tj. entitet.

Skup promenljivih koje su potrebne da bi se sistem opisao u datom trenutku vremena, a prema usvojenom cilju posmatranja, naziva se *stanjem* toga sistema. U primeru ulaznog reda čekanja, promenljive stanja bile bi broj, veličine i trenuci prispeća paketa u njegovom baferu.

Sistem čije promenljive stanja menjaju svoje vrednosti skokovito, u diskretnim trenucima vremena je *diskretan*. Nasuprot tome, sistem čije se promenljive stanja menjaju kontinualno u vremenu je *kontinualan*. Primetimo da su u stvarnosti malobrojni sistemi koji su potpuno diskretni ili kontinualni; pripadnost sistema jednoj od ovih dveju kategorija stoga se određuje prema *dominantnom* načinu promene vrednosti promenljivih stanja.

Razmotrimo sada moguće načine za analizu sistema, koji su ilustrovani na slici 1.1.



Slika 1.1: Metode za analizu sistema.

Idealno bi bilo kada bismo eksperiment mogli izvršiti nad *stvarnim sistemom*, jer bismo tada dobili najverodostojnije podatke o njegovom ponašanju. U stvarnosti ovakav pristup često nije primenljiv; zadržimo li se na primeru rutera, neki od mogućih razloga za to bili bi:

- ne možemo priuštiti ruter nad kojim bismo eksperimentisali zbog njegove visoke cene,
- ne možemo menjati konfiguraciju rutera u „živoj" mreži zbog opasnosti da bi se time narušio njen regularan režim rada,
- ne postoji fizički ruter nad kojim bismo mogli izvršiti eksperiment, jer npr. razvijamo novi protokol, koji još uvek nije implementiran.

Ukoliko se za trenutak prebacimo iz domena telekomunikacija u npr. elektroenergetske sisteme i, saglasno tome, umesto rutera posmatramo npr. nuklearnu elektranu, razlozi protiv ovoga pristupa biće još upečatljiviji.

Kada nam nije dostupan stvarni sistem, alternativa koja nam preostaje je eksperiment nad njegovim surogatom u vidu *modela*.

Model može biti fizički i matematički. *Fizički model* predstavlja umanjenu varijantu stvarnog sistema; studenti elektrotehnike susreću se sa ovakvim modelima u vidu maketa za laboratorijske vežbe. Dobra strana fizičkih modela je njihova očiglednost, zbog čega se prvenstveno i koriste u obrazovne svrhe; mane su im cena i ograničene mogućnosti primene. Nasuprot njima, *matematički modeli* predstavljaju skup kvantitativnih i logičkih zavisnosti čijim se rešavanjem može doći do odziva sistema pod datim okolnostima. Zbog svoje skalabilnosti, matematički modeli su daleko značajniji za praktične

primene, ali im je mana to što po pravilu u opis sistema uvode *pojednostavljenja* u vidu aproksimacija i zanemarivanja. Pitanjem kvalitetnog modeliranja telekomunikacionih sistema detaljnije ćemo se baviti u poglavlju 2.

Matematički modeli mogu biti *diskretni* i *kontinualni*, u zavisnosti od toga da li se odnose na diskretne ili na kontinualne sisteme, respektivno. Ukoliko u matematičkom modelu eksplicitno ne figuriše vreme kao promenljiva – bilo zbog toga što se model odnosi na opis sistema u datom trenutku vremena, ili zato što je sistem vremenski invarijantan – on je *statički*; nasuprot tome, *dinamički model* pokazuje evoluciju sistema u vremenu. Konačno, ako u modelu ne figurišu slučajne (probabilističke) veličine, on je *deterministički*, dok se u suprotnom radi o *stohastičkom modelu*.

Nezavisno od njegovog konkretnog tipa, ako je matematički model jednostavan, rešićemo ga *analitički*. Po nepisanom pravilu, iole detaljniji matematički modeli ili nemaju analitičko rešenje, ili je ono toliko komplikovano ili nezgrapno da nije pogodno za praktičan rad. U ovakvim situacijama, metod izbora za analizu sistema je računarska *simulacija*, tj. numeričko rešavanje njegovog matematičkog modela, da bi se *ocenio*[1] odziv „originalnog" fizičkog sistema. Matematički model koji se rešava simulacijom naziva se i *simulacionim modelom*.

Činjenica da se rezultat simulacije može tumačiti samo kao ocena parametara sistema, pogotovo onda kada njegov model ima stohastičku komponentu, tj. kada u njemu figurišu slučajni procesi, predstavlja i najveći metodološki nedostatak simulacije. Njene prednosti su, s druge strane, višestruke. Već smo kazali da simulacija omogućava rešavanje problema koji nemaju analitičko rešenje. Dalje, simulacija omogućava *neinvazivnu* analizu postojećih sistema, kao i analizu još uvek nepostojećih sistema, koji se nalaze u fazi projektovanja i razvoja; u potonjem slučaju, simulacija se često koristi za optimizaciju, u smislu poređenja alternativnih konfiguracija sistema. Simulacioni eksperiment pruža veću slobodu u izboru parametara, a takođe se ima i veća kontrola nad samim eksperimentom. Na kraju, simulacija omogućava „razvlačenje skale" vremena pri posmatranju sistema; spori procesi tako se mogu posmatrati ubrzano, dok se brzi mogu usporiti.

Izlišno je napominjati da simulacija nije vezana isključivo za telekomunikacije. Ona se primenjuje u svim tehničkim i prirodnim naukama, kao i u ekonomiji. U popularnoj kulturi se simulacija pogrešno vezuje za svaku primenu računara za rešavanje nekog problema. Iako podrazumeva izvođenje numeričkog eksperimenta nad matematičkim modelom sistema pomoću računara, simulacija *nije* primena računara za puko ilustrovanje analitičkih zavisnosti, bilo kroz crtanje njihovih grafika, ili kroz animaciju. Simulacija *nije* ni puko programiranje u smislu izvršavanja algoritama, jer pored veštine programiranja podrazumeva i dobro poznavanje sistema koji se analizira, kao i matematike, naročito numeričkih metoda i statistike.

Nakon što smo razjasnili osnovne pojmove, spremni smo da se upustimo u detaljnije izučavanje modeliranja i simulacije u telekomunikacijama.

[1] „Ocena parametara" je uobičajen izraz u matematici, pa ćemo ga i mi koristiti umesto lingvistički podesnijeg izraza „procena".

Poglavlje 2

Modeliranje procesa u telekomunikacionim sistemima

U ovome poglavlju ćemo razmotriti formiranje matematičkih modela procesa u telekomunikacionim sistemima. Cilj nam je da dobijemo model koji će što vernije odražavati fizičku realnost, ali i istovremeno biti pogodan za računarsku implementaciju.

Od čitalaca se očekuje da su upoznati s osnovama telekomunikacija i digitalne obrade signala; stoga ćemo nakon uvoda odmah preći na neke konkretne probleme.

2.1 Metodologija modeliranja

Neka je ponašanje nekog sistema opisano ulazno-izlaznom relacijom

$$\mathbf{Y} = f(\mathbf{X}). \tag{2.1}$$

$\mathbf{X}$ je vektor ulaznih signala, $\mathbf{Y}$ vektor izlaznih signala, dok funkcija f predstavlja model sistema.

Podsetimo se: Ako model ne zavisi od trenutka posmatranja, kaže se da je statički ili vremenski invarijantan; u suprotnom, model je dinamički. Model je deterministički ukoliko ne sadrži slučajne promenljive, dok je u suprotnom stohastički. Konačno, u zavisnosti od toga koju vrstu procesa opisuje, model može biti kontinualan ili diskretan.

Zadatak modeliranja je da se odredi analitički oblik zavisnosti f. Kao što smo videli, to nije uvek moguće, pa u takvim slučajevima pribegavamo pojednostavljenju originalnog problema. Jedan način za to je da umesto analitički složene ili nepoznate zavisnosti f posmatramo njenu aproksimaciju $\hat{f}$; na taj način se originalni sistem aproksimira pojednostavljenim,

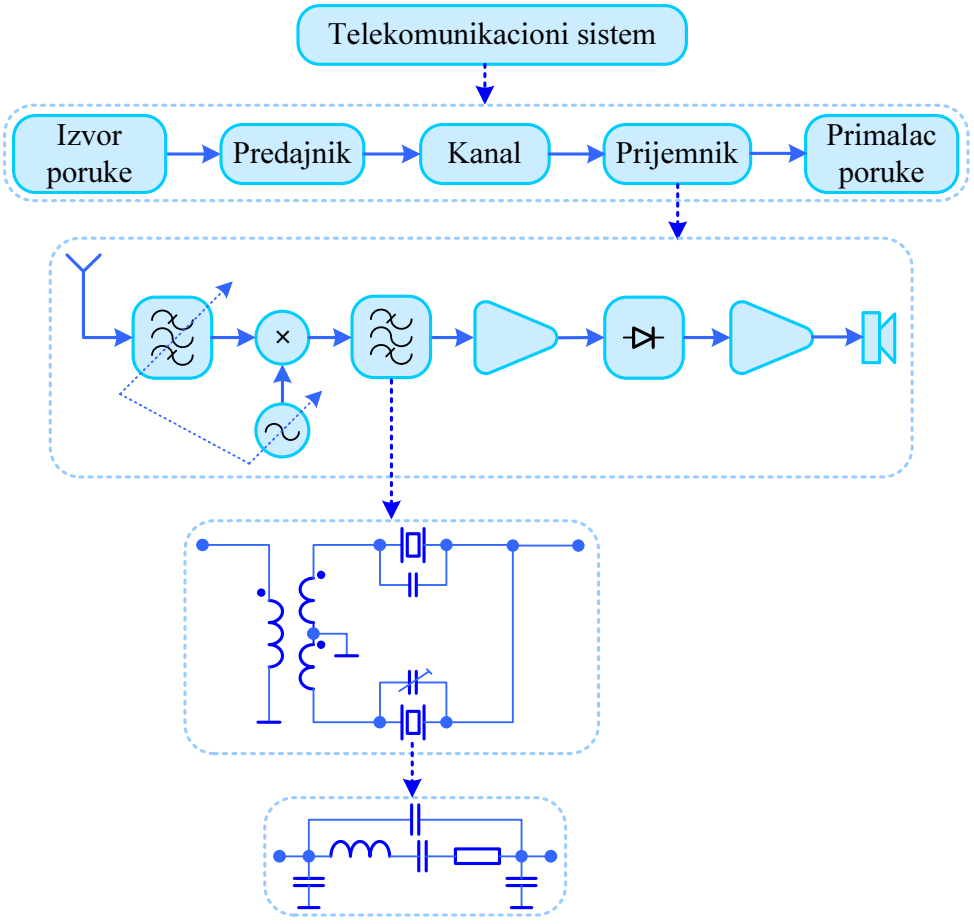
$$\mathbf{Y} = \hat{f}(\mathbf{X}). \tag{2.2}$$

Drugi pristup podrazumeva da se ne posmatraju svi ulazni signali, već da se neki od njih zamene konstantama; tako se sistem umesto vektorom $\mathbf{X} = [x_1, x_2, \ldots, x_n]$ pobuđuje novim vektorom $\hat{\mathbf{X}} = [x_1, x_2, \ldots, x_k, c_1, c_2, \ldots, c_{n-k}]$. Ukoliko su konstante c_i, $i = 1, \ldots, n-k$ jednake nulama, ovim se zanemaruju uticaji pridruženih ulaznih signala.

Moguće je i kombinovati ova dva pristupa, što znači da se posmatra sistem

$$\mathbf{Y} = \hat{f}(\hat{\mathbf{X}}). \tag{2.3}$$

Pri formiranju modela ne treba voditi računa samo o preciznosti kojom se opisuje posmatrani sistem (ili pojava), već i o računskoj složenosti njegove implementacije. Kvalitet modela se ogleda u postizanju kompromisa između ovih dvaju oprečnih zahteva. Ilustrovaćemo to na primeru koji je prikazan na slici 2.1.



Slika 2.1: Hijerarhija modela telekomunikacionog sistema.

Posmatrajmo jedan telekomunikacioni sistem. Na prvom nivou modeliranja, taj sistem možemo predstaviti blok-dijagramom koji se sastoji od izvora poruke, predajnika, kanala, prijemnika i primaoca poruke. Na ovome nivou ne ulazimo u strukturu pojedinih blokova, već ih opisujemo fenomenološki, kao „crne kutije".

Na drugom nivou detaljnije razrađujemo strukturu pojedinih gradivnih blokova, ali i dalje zadržavamo fenomenološki pristup. Na primer, pretpostavljamo da je prijemnik superheterodinskog tipa, izgrađen od standardnih sklopova koji su opisani ulazno-izlaznim relacijama, impulsnim odzivom ili funkcijom prenosa.

Na trećem nivou se posmatraju realizacije pojedinih sklopova prijemnika na nivou električne šeme; na slici je tako nacrtana šema međufrekvencijskog filtra u kome su primenjene jedinke kristala kvarca. Pretpostavljamo da su sve komponente idealnih karakteristika.

Ekvivalente šeme pojedinih komponenti se posmatraju na četvrtom nivou. Na slici je tako data ekvivalentna šema kvarcne jedinke, na kojoj su ucrtane i parazitne kapacitivnosti izvoda prema masi.

Na još višim nivoima modeliranja bi se posmatrala implementacija pojedinih komponenti (npr. rezovi kristala) i fizički procesi u njima (kretanje nosilaca naelektrisanja, struktura elektromagnetskog polja i sl).

Iz ovoga primera je očigledno da se s povećavanjem nivoa modela sistem vernije opisuje, ali i da taj opis postaje matematički sve složeniji. U telekomunikacijama se najčešće primenjuje modeliranje do drugog nivoa. Treći i četvrti nivoi modeliranja su uobičajeni u elektronici, dok se modeli višeg nivoa koriste pri projektovanju komponenti.

2.2 Niskofrekvencijski ekvivalenti telekomunikacionih signala

Posmatrajmo modulisani signal dat izrazom

$$x(t) = r(t)\cos\left[2\pi f_c t + \varphi(t)\right], \tag{2.4}$$

u kome je sa $r(t)$ označena promenljiva amplituda, sa f_c frekvencija nosioca, a sa $\varphi(t)$ devijacija faze. Ovaj izraz možemo napisati u obliku

$$x(t) = \mathrm{Re}\left\{r(t)e^{j\varphi(t)}e^{j2\pi f_c t}\right\}. \tag{2.5}$$

Kompleksna anvelopa ili kompleksni niskofrekvencijski ekvivalent signala $x(t)$ definiše se kao

$$\tilde{x}(t) = r(t)e^{j\varphi(t)}. \tag{2.6}$$

Ukoliko opseg koji zauzima spektar signala $x(t)$, B, zadovoljava uslov $B \ll f_c$, tada se obrada modulisanog signala na frekvenciji f_c može ekvivalentirati obradom njegove kompleksne anvelope na niskim frekvencijama.

Formirajmo tzv. analitički signal signala $x(t)$,

$$z_x(t) = x(t) + jx_H(t), \tag{2.7}$$

gde je sa $x_H(t)$ označena Hilbertova transformacija signala $x(t)$:

$$x_H(t) = \mathcal{H}\{x(t)\} = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{x(\tau)}{t-\tau}\, \mathrm{d}\tau. \tag{2.8}$$

Nije teško videti da važi

$$x_H(t) = \mathcal{F}^{-1}\{-jX(f)\,\mathrm{sgn}(f)\} = x(t) * \frac{1}{\pi t}, \tag{2.9}$$

gde je simbolom $*$ označena operacija konvolucije. Izračunavanje Hilbertove transformacije stoga možemo shvatiti kao proces filtriranja signala kroz tzv. Hilbertov transformator, kvadraturni filtar svepropusnik frekvencija, koji samo unosi fazni pomeraj od $\pi/2$. Njegov impulsni odziv je

$$h(t) = \frac{1}{\pi t}, \tag{2.10}$$

a funkcija prenosa

$$H(f) = -j\,\mathrm{sgn}(f). \tag{2.11}$$

Veza analitičkog signala i kompleksne anvelope je

$$z_x(t) = \tilde{x}(t)\, e^{j2\pi f_c t}. \tag{2.12}$$

Iz definicije Hilbertove transformacije sledi njena važna osobina:

$$\mathcal{H}\{x_H(t)\} = -x(t), \tag{2.13}$$

a takođe je

$$\mathcal{F}\{x_H(t)\} = X_H(f) = -jX(f)\,\mathrm{sgn}(f), \tag{2.14}$$

kao i

$$\mathcal{H}\{\cos(2\pi f_c t)\} = \sin(2\pi f_c t)\,, \tag{2.15}$$

$$\mathcal{H}\{\sin(2\pi f_c t)\} = -\cos(2\pi f_c t)\,. \tag{2.16}$$

Ako signal $a(t)$ ima ograničen spektar, $|A(f)| = 0$, $|f| \geqslant B$, tada će važiti

$$\mathcal{H}\{a(t)\cos(2\pi f_c t)\} = a(t)\sin(2\pi f_c t)\,, \tag{2.17}$$

$$\mathcal{H}\{a(t)\sin(2\pi f_c t)\} = -a(t)\cos(2\pi f_c t)\,, \tag{2.18}$$

pa Hilbertova transformacija u ovome slučaju ne deluje na modulišući signal, već samo na nosilac. Slabiji uslov da bi ovo važilo je $f_c \geqslant B$.

Razmotrimo sada neke praktične aspekte rada s niskofrekvencijskim ekvivalentom signala. Modulisani signal $x(t)$, dat izrazom (2.4), možemo napisati u ekvivalentnom obliku

$$x(t) = p(t)\cos(2\pi f_c t) - q(t)\sin(2\pi f_c t)\,, \tag{2.19}$$

gde su $p(t)$ i $q(t)$ redom komponenta u fazi i komponenta u kvadraturi, ograničenog spektra. Hilbertova transformacija signala $x(t)$ sada će biti

$$x_H(t) = p(t)\sin(2\pi f_c t) + q(t)\cos(2\pi f_c t)\,, \tag{2.20}$$

pa je njegov analitički signal

$$z_x(t) = [p(t) + jq(t)]\, e^{j2\pi f_c t}. \tag{2.21}$$

Prema (2.12), kompleksna anvelopa signala $x(t)$ odavde se može napisati u obliku

$$\tilde{x}(t) = p(t) + jq(t). \tag{2.22}$$

Vidimo da se komponenta u fazi direktno preslikava u realni, a komponenta u kvadraturi u imaginarni deo kompleksne anvelope. Spektar kompleksne anvelope sada je

$$\tilde{X}(f) = P(f) + jQ(f), \tag{2.23}$$

dok su komponente u fazi i kvadraturi povezane s promenljivom amplitudom i devijacijom faze izrazima

$$r(t) = \sqrt{p^2(t) + q^2(t)}, \tag{2.24}$$

$$\varphi(t) = \operatorname{arctg}\frac{q(t)}{p(t)}. \tag{2.25}$$

Po cenu nešto duže pripreme, posmatranjem niskofrekvencijskog ekvivalenta moguće je smanjiti potrebnu frekvenciju odabiranja, pa će za obradu signala trebati manje odbiraka; ovo će za posledicu imati manje zauzeće memorije i manje opterećenje procesora računara, pa će se simulacija izvršavati brže.

Da bi model preko niskofrekvencijskog ekvivalenta verno odražavao stvarni sistem, ne treba zaboraviti pod kojim smo ga pretpostavkama izveli – modulišući signal je ograničenog spektra, čija je širina manja od frekvencije nosioca. Primeri sistema u kojima ove pretpostavke nisu ispunjene su sistemi s frekvencijskom modulacijom i telefonski (*voice-band*) modemi.

2.3 Niskofrekvencijski ekvivalenti filtara

Uobičajeno je da se filtri projektuju kao normalizovani niskoferkvencijski prototipi i potom transformišu u željenu konfiguraciju.

Posmatrajmo filtar propusnik opsega frekvencija, čije su granične kružne učestanosti propusnog opsega redom ω_L i ω_H. Širina njegovog propusnog opsega je $\omega_b = \omega_H - \omega_L$, dok je centralna kružna učestanost propusnog opsega

$$\omega_0 = \frac{\omega_L + \omega_H}{2}. \tag{2.26}$$

Definišimo dalje geometrijsku centralnu učestanost propusnog opsega,

$$\omega_g = \sqrt{\omega_L \omega_H}. \tag{2.27}$$

Za filtre propusnike ili nepropusnike opsega, tipično je $\omega_g \gg \omega_b$.

Imajući u vidu definicione izraze parametara, sada možemo pisati

$$\omega_L = \omega_0 - \frac{\omega_b}{2}, \tag{2.28}$$

$$\omega_H = \omega_0 + \frac{\omega_b}{2} \tag{2.29}$$

i

$$\omega_g = \sqrt{\omega_0^2 - \left(\frac{\omega_b}{2}\right)^2} \approx \omega_0. \tag{2.30}$$

Odavde zaključujemo da će, ako je prenosna funkcija filtra simetrična oko centralne frekvencije propusnog ili, u zavisnosti od tipa filtra, nepropusnog opsega, njegov niskofrekvencijski ekvivalent odgovarati niskofrekvencijskom prototipu, koji je centriran oko nulte učestanosti i čija je širina propusnog opsega $\omega_b/2$.

Transformacije filtara u niskofrekvencijske prototipe date su u tabeli 2.1.

Tabela 2.1: Transformacije filtara u NF prototip.

Tip filtra	Smena
Propusnik niskih frekvencija	$\frac{s}{\omega_b} \to s$
Propusnik visokih frekvencija	$\frac{\omega_b}{s} \to s$
Propusnik opsega	$\frac{s^2+\omega_g^2}{\omega_b s} \to s$
Nepropusnik opsega	$\frac{\omega_b s}{s^2+\omega_g^2} \to s$

Ako nije poznat niskofrekvencijski prototip filtra, njegov niskofrekvencijski ekvivalent ćemo približno odrediti na sledeći način:

1. funkciju prenosa ćemo predstaviti u vidu zbira parcijalnih razlomaka,
2. odbacićemo nule i polove koji odgovaraju negativnim vrednostima kružne učestanosti ω i
3. preostale nule i polove ćemo translirati duž ordinate, za vrednost centralne kružne učestanosti propusnog/nepropusnog opsega.

Na primer, neka je zadat filtar funkcije prenosa

$$H(s) = \frac{2\left(s^2 + 5s - 2494\right)}{\left(s^2 + 4s + 2504\right)\left(s^2 + 6s + 2509\right)},$$

čija je centralna kružna učestanost propusnog opsega $\omega_0 = 2\pi f_0 = 50$ rad/s. Razvoj funkcije prenosa u parcijalne razlomke je

$$H(s) = \frac{1}{s + 2 + j50} + \frac{1}{s + 2 - j50} - \frac{1}{s + 3 + j50} - \frac{1}{s + 3 - j50}.$$

Odbacujemo prvi i treći razlomak, pa će biti

$$H_p(s) = \frac{1}{s+2-j50} - \frac{1}{s+3-j50}.$$

Smenom $s \leftarrow s - j50$ dobijamo traženu funkciju prenosa niskofrekvencijskog ekvivalenta:

$$H_L(s) = \frac{1}{s+2} - \frac{1}{s+3} = \frac{1}{s^2+5s+6}.$$

2.4 Neke važne raspodele

U ovome odeljku ćemo se podsetiti nekih raspodela koje imaju primenu u modeliranju telekomunikacionih sistema.

2.4.1 Diskretne raspodele

Bernoullijeva raspodela

Prva raspodela koju ćemo obraditi je Bernoullijeva. Ovde slučajna promenljiva ima dve moguće vrednosti, koje redom uzima s verovatnoćama p i $q = 1 - p$.

Bernoullijeva slučajna promenljiva predstavlja model statističkih eksperimenata koji imaju dva ishoda (tzv. Bernoullijevih eksperimenata). U telekomunikacijama se koristi kao indikator događaja, kao što su, na primer, greška bita, uspešan prijem paketa, zauzeće linka i sl.

Uniformna raspodela

U diskretnoj uniformnoj raspodeli postoji n mogućih ishoda statističkog eksperimenta, koje slučajna promenljiva X uzima s jednakim verovatoćama:

$$\mathrm{P}\,(X = k) = \frac{1}{n}\,, \quad k = 1, 2, \ldots, n. \tag{2.31}$$

Ovakvom raspodelom se na primer opisuje generisanje slučajnih binarnih (za $n = 2$) ili M-arnih (za $n = M$) sekvenci. Uobičajena oznaka da slučajna promenljiva X pripada diskretnoj uniformnoj raspodeli je $X \sim \mathrm{Unif}\,(n)$.

Binomna raspodela

Ako slučajna promenljiva X odgovara broju uspeha u n nezavisnih Bernoullijevih eksperimenata (npr. slanja paketa), pri čemu je verovatnoća pozitivnog ishoda jednog

eksperimenta (verovatnoća uspeha u jednom pokušaju) p, njena raspodela će biti binomna:

$$\mathrm{P}\left(X=k\right)=\binom{n}{k}p^k(1-p)^{n-k}\,,\quad k=0,1,2,\ldots,n. \tag{2.32}$$

Oznaka za binomnu raspodelu je $\mathrm{Bin}(n,p)$. Za $n=1$, binomna raspodela se svodi na Bernoullijevu.

Negativna binomna raspodela

Ukoliko se nezavisni Bernoullijevi eksperimenti budu ponavljali sve dok se ne bude ostvario r-ti uspeh (npr. dok se ne bude uspešno prenelo r paketa), broj pokušaja će imati negativnu binomnu ili Pascalovu raspodelu:

$$\mathrm{P}\left(X=k\right)=\binom{k-1}{r-1}p^r(1-p)^{k-r}\,,\quad k=r,r+1,\ldots \tag{2.33}$$

Raspodela koja se dobije za $r=1$ (broj pokušaja do prvog uspeha) naziva se geometrijskom.

Poissonova raspodela

Poissonova raspodela opisuje tzv. retke događaje, kao što su, pod određenim okolnostima, generisanje poruka ili telefonskih poziva. Slučajna promenljiva X ovde odgovara broju događaja u jediničnom intervalu vremena, pod sledećim pretpostavkama:

- u datom trenutku se javlja najviše jedan događaj,
- svaki interval konačnog trajanja sadrži konačan broj događaja, a svaki interval beskonačnog trajanja sadrži beskonačan broj događaja,
- događaji se javljaju u slučajnim trenucima vremena i
- brojevi događaja u nepreklapajućim intervalima vremena su nezavisni.

Tada je

$$\mathrm{P}\left(X=k\right)=\frac{\lambda^k}{k!}\,e^{-\lambda}\,,\quad k=0,1,\ldots \tag{2.34}$$

Oznaka da slučajna promenljiva X ima Poissonovu raspodelu je $X\sim\mathrm{Poiss}(\lambda)$.

2.4.2 Kontinualne raspodele

Uniformna raspodela

Kontinualna uniformna raspodela je opisana funkcijom gustine verovatnoće

$$f_X(x)=\frac{1}{b-a}\,,\quad a<x<b. \tag{2.35}$$

Videćemo da ova raspodela ima značajnu primenu u generisanju slučajnih brojeva. Simbolička oznaka za nju je Unif(a, b).

Gaussova raspodela

Funkcija gustine verovatnoće Gaussove ili normalne raspodele je

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad -\infty < x < +\infty. \tag{2.36}$$

Oznaka za ovu raspodelu je $\mathcal{N}(\mu, \sigma^2)$. Raspodela $\mathcal{N}(0, 1)$ se naziva standardnom normalnom raspodelom.

Gaussovom raspodelom se npr. opisuje termički šum.

Eksponencijalna raspodela

Eksponencijalna raspodela je data funkcijom gustine verovatnoće

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0. \tag{2.37}$$

Simbolička oznaka za eksponencijalnu raspodelu je Exp(λ).

Među eksponencijalnom i Poissonovom raspodelom postoji zanimljiva veza: vremena između sukcesivnih Poissonovih događaja (tzv. vremena međudolazaka) imaju eksponencijalnu raspodelu.

Gama raspodela

Funkcija gustine verovatnoće gama raspodele je

$$f_X(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}, \quad 0 < x < \infty;\ \alpha, \beta > 0. \tag{2.38}$$

U gornjem izrazu, $\Gamma(\alpha)$ je gama funkcija:

$$\Gamma(\alpha) = \int_0^\infty y^{\alpha-1} e^{-y}\, \mathrm{d}y. \tag{2.39}$$

Pošto je $\Gamma(\alpha) = (\alpha - 1)\Gamma(\alpha - 1)$, kaže se da gama funkcija predstavlja generalizaciju faktorijela.

Oznaka za gama raspodelu je $\gamma(\alpha, \beta)$.

Ako slučajna promenljiva X predstavlja vremenski interval potreban za α realizacija Poissonovog procesa parametra λ, imaće raspodelu $\gamma(\alpha, 1/\lambda)$. Za $\alpha = 1$, gama raspodela se svodi na eksponencijalnu, $\mathrm{Exp}(1/\beta)$. Ako je $\alpha = k \in \mathbb{N}$, dobijena raspodela se naziva Erlangovom:

$$f_X(x) = \frac{\lambda^k}{(k-1)!}\, x^{k-1} e^{-\lambda x}, \quad 0 < x < \infty;\ k, \lambda > 0. \tag{2.40}$$

Rayleighjeva raspodela

Ako su X_1 i X_2 nezavisne slučajne promenljive iz raspodele $\mathcal{N}(\mu, \sigma^2)$, slučajna promenljiva

$$X = \sqrt{X_1^2 + X_2^2} \tag{2.41}$$

imaće Rayleighjevu raspodelu:

$$f_X(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right), \quad x > 0. \tag{2.42}$$

Rayleighjevom raspodelom se opisuju amplituda uskopojasnog šuma i jedna klasa fedinga.

2.5 Transformacije slučajnih promenljivih

Neka je zadata kontinualna slučajna promenljiva X s poznatom funkcijom gustine verovatnoće $f_X(x)$. Konstruišimo novu slučajnu promenljivu $Y = g(X)$, tako da postoji $k \geqslant 1$ inverzija $x^{(i)} = g_i^{-1}(y)$. Funkcija gustine raspodele slučajne promenljive Y tada je data izrazom

$$f_Y(y) = \sum_{i=1}^{k} \frac{f_X\left(x^{(i)}\right)}{\left|g'\left(x^{(i)}\right)\right|}. \tag{2.43}$$

Ako se radi o slučajnom vektoru koji je funkcija drugog slučajnog vektora, tj. ako je $Y_i = g_i(X_1, X_2, \ldots, X_n)$, $i = 1, 2, \ldots, n$, funkcija gustine verovatnoće će biti

$$f_{\mathbf{Y}}(\mathbf{y}) = \sum_{i=1}^{k} \frac{f_{\mathbf{X}}(\mathbf{x}^{(i)})}{|J(\mathbf{x}^{(i)})|}, \tag{2.44}$$

gde je sa $J(\mathbf{x}^{(i)})$ označen Jacobijan transformacije:

$$J(\mathbf{x}^{(i)}) = \left| \begin{array}{cccc} \dfrac{\partial g_1}{\partial x_1} & \dfrac{\partial g_1}{\partial x_2} & \cdots & \dfrac{\partial g_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial g_n}{\partial x_1} & \dfrac{\partial g_n}{\partial x_2} & \cdots & \dfrac{\partial g_n}{\partial x_n} \end{array} \right|_{\mathbf{x}=\mathbf{x}^{(i)}}. \tag{2.45}$$

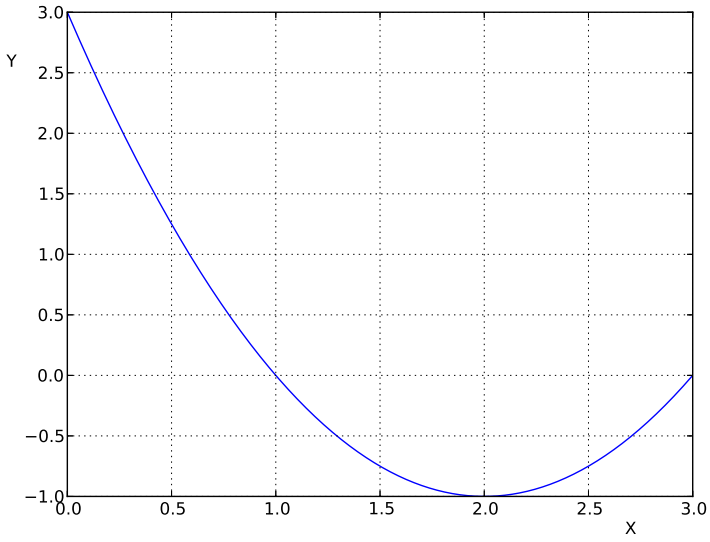
Ilustrujmo ove izraze na primerima.

Primer 1

Neka je, za početak, $X \sim \text{Unif}\,[0,3]$ i $Y = X^2 - 4X + 3$. Funkcija gustine verovatnoće X je

$$f_X(x) = \begin{cases} \frac{1}{3}, & x \in [0,3] \\ 0, & \text{inače} \end{cases}.$$

Grafik funkcije $Y = g(X)$ je prikazan na slici.



Slika 2.2: Grafik preslikavanja.

Na intervalu $Y \in [-1,0)$ postoje dve inverzije $X = 2 \pm \sqrt{(1+Y)}$, dok na intervalu $Y \in [0,3]$ postoji samo jedna inverzija, $X = 2 - \sqrt{(1+Y)}$. Izvod preslikavanja je $Y' = 2X - 4$. Stoga će biti

$$f_Y(y) = \begin{cases} \dfrac{1}{3\sqrt{1+Y}}, & y \in [-1,0) \\ \dfrac{1}{6\sqrt{1+Y}}, & y \in [0,3] \\ 0, & \text{inače} \end{cases}.$$

Proverom dobijamo da je zaista

$$\int_{-1}^{3} f_Y(y)\,\mathrm{d}y = 1.$$

Primer 2

Kao drugi primer, neka je $Y = \sin X$ i $X \sim \text{Unif}\,[0, 2\pi]$. Inverzija $X = \arcsin Y$ definisana je na intervalu $Y \in [-\pi/2, \pi/2]$. Na intervalu $X \in [0, 2\pi]$ biće

$$X = \begin{cases} \arcsin Y \text{ ili } \pi - \arcsin Y, & Y \in [0, 1] \\ 2\pi + \arcsin Y \text{ ili } \pi - \arcsin Y, & Y \in [-1, 0) \end{cases}.$$

Izvod transformacije je $Y' = \cos X = \sqrt{1 - \sin^2 X}$ i za sve inverzije iznosi $\sqrt{1 - Y^2}$. Stoga je

$$f_Y(y) = \frac{1}{\pi\sqrt{1 - y^2}}, \quad -1 < y < 1.$$

Proverimo i ovaj rezultat:

$$\int_{-1}^{1} f_Y(y)\,\mathrm{d}y = \frac{1}{\pi}\arcsin y\Big|_{-1}^{1} = 1.$$

Primer 3

Neka je, u trećem primeru, Y funkcija slučajnog vektora:

$$Y = X_1 + X_2,$$

gde su X_1 i X_2 nezavisne identično raspodeljene[1] slučajne promenljive. Njihova združena funkcija gustine verovatnoće stoga je

$$f_{X_1,X_2}(x_1, x_2) = f_{X_1}(x_1) f_{X_2}(x_2).$$

Da bismo mogli da primenimo izraz (2.44), neophodno je da se podudaraju dimenzije slučajnih vektora $\mathbf{X}$ i $\mathbf{Y}$. Stoga ćemo formalno definisati

$$Y_1 = X_1 + X_2$$

i

$$Y_2 = X_2.$$

Inverzije ovoga sistema su

$$X_1 = Y_1 - Y_2,$$
$$X_2 = Y_2,$$

pa je Jacobijan transformacije

$$J = \begin{vmatrix} 1 & -1 \\ 0 & 1 \end{vmatrix} = 1.$$

[1] U literaturi se koristi oznaka iid, što potiče od engleskog izraza *independent identically distributed*.

Združena funkcija gustine verovatnoće stoga je

$$f_{Y_1,Y_2}(y_1,y_2) = \frac{f_{X_1}(y_1-y_2)f_{X_2}(y_2)}{1}.$$

Marginalnu funkciju gustine verovatnoće za Y_1 dobijamo njenim integriranjem:

$$f_{Y_1}(y_1) = \int_{-\infty}^{+\infty} f_{Y_1,Y_2}(y_1,y_2)\,\mathrm{d}y_2 = \int_{-\infty}^{+\infty} f_{X_1}(y_1-y_2)f_{X_2}(y_2)\,\mathrm{d}y_2,$$

pa je funkcija gustine verovatnoće zbira iid slučajnih promenljivih jednaka konvoluciji njihovih funkcija gustine verovatnoća.

2.6 Centralna granična teorema

Neka su $X_1, X_2, \ldots, X_n$ iid slučajne promenljive, čije je matematičko očekivanje μ i varijansa σ^2. Formirajmo novu slučajnu promenljivu

$$Z_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \frac{X_i - \mu}{\sigma}, \quad n = 1, 2, \ldots \tag{2.46}$$

Tada će važiti

$$(\forall z) \quad \lim_{n\to\infty} \mathrm{P}\left(Z_n \leqslant z\right) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}}\, e^{-y^2/2}\,\mathrm{d}y, \tag{2.47}$$

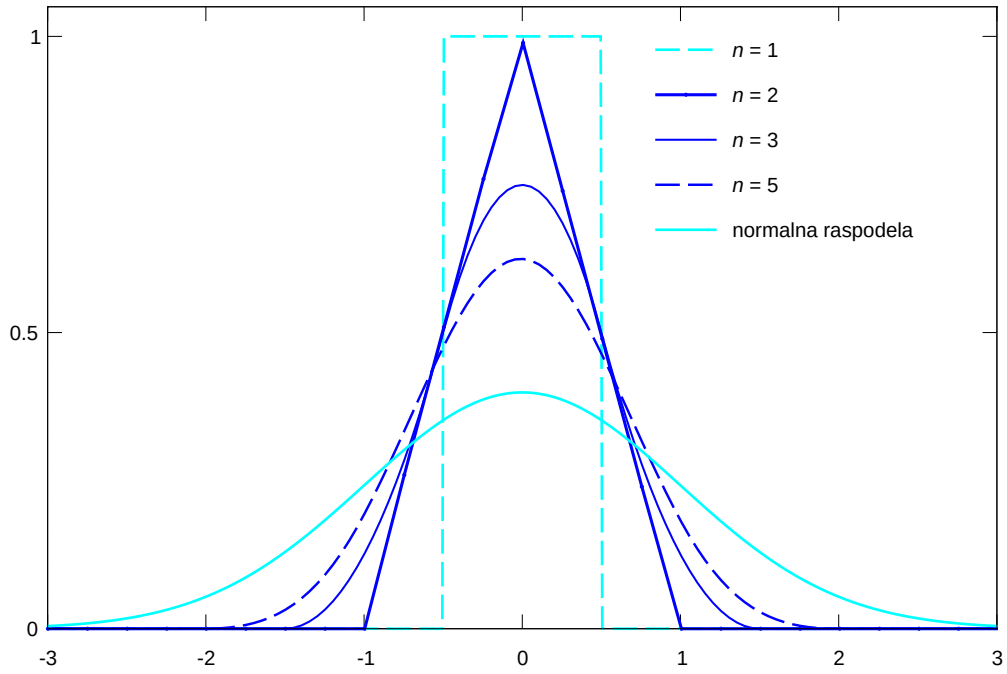
pa će Z_n konvergirati u verovatnoći ka standardnoj normalnoj slučajnoj promenljivoj. Ova relacija se naziva centralnom graničnom teoremom (CGT) i, za slučaj $X_i \sim \mathrm{Unif}\,[0,1]$, ilustrovana je na slici 2.3.

2.7 Slučajni procesi

Slučajni proces predstavlja preslikavanje ishoda statističkog eksperimenta u funkciju vremena.

Slučajni proces je stacionaran u užem smislu ako njegova raspodela ne zavisi od trenutka posmatranja. Slučajan proces je stacionaran u širem smislu ako je $\mathrm{E}X(t) = const.$ i ako $\mathrm{E}X(t_1)X(t_2)$ ne zavisi eksplicitno od t_1 i t_2, već samo od njihove razlike.

Slučajni proces je ergodičan (ili ergodski) po srednjoj vrednosti ako su mu srednje vrednosti po vremenu jednake srednjim vrednostima po realizacijama. Proces može biti ergodičan i po momentima višeg reda, kao i po drugim numeričkim karakteristikama; zajedničko za sve varijante ergodičnosti je da se numeričke karakteristike mogu izračunavati na osnovu samo jedne vremenske realizacije procesa.



Slika 2.3: Konvergencija funkcije gustine verovatnoće sume nezavisnih uniformno raspodeljenih slučajnih promenljivih ka normalnoj raspodeli.

2.7.1 Primeri slučajnih procesa u telekomunikacijama

U simulaciji, slučajni procesi služe kao matematički modeli stohastičkih pojava. Termički šum se tako modelira Gaussovim slučajnim procesom. Impulsni šumovi, kao što je to npr. šum sačme, modeliraju se procesom oblika

$$X(t) = \sum_{k=-\infty}^{+\infty} h(t - \tau_k), \tag{2.48}$$

gde je $h(t)$ vremenski oblik jednog impulsa i τ Poissonova slučajna promenljiva.

Beli šum se modelira nezavisnom sekvencom $X(n)$, za koju važi: $(\forall k)$ i $(\forall j \neq 0)$, $X(k)$ i $X(k+j)$ su nezavisne slučajne promenljive, takve da je:

$$\mathrm{E}X(n) = \mu, \tag{2.49}$$

$$\mathrm{E}X(n)X(n+j) = \begin{cases} \sigma^2, & j = 0 \\ 0, & j \neq 0 \end{cases}. \tag{2.50}$$

Sa σ^2 pri tome je označena jednostrana spektralna gustina snage procesa, za opseg frekvencija $|f| < f_s/2$, gde je f_s frekvencija odabiranja.

Odbirci signala govora i videa, sukcesivna slova u književnom jeziku, generisanje blokovskih grešaka i sl. modeliraju se markovskim procesima k-tog reda, za koje važi

$$\begin{aligned} &\mathrm{P}\left[X(n)|X(n-1), X(n-2), \ldots, X(n-m)\right] = \\ &= \mathrm{P}\left[X(n)|X(n-1), X(n-2), \ldots, X(n-k)\right], \quad k < m. \end{aligned} \tag{2.51}$$

ARMA (*autoregressive moving average*) procesi imaju značaj u modeliranju telekomunikacionog saobraćaja. Oni su definisani izrazom

$$X(n) = \sum_{i=1}^{p} \alpha_{p,i} X(n-i) + \sum_{k=1}^{q} \theta_{q,k} e(n-k) + e(n). \tag{2.52}$$

Pri tome je $e(n)$ stacionarna Gaussova sekvenca, nultog očekivanja i varijanse σ^2. Alternativni zapis ARMA procesa je

$$\left(1 - \sum_{i=1}^{p} \alpha_{p,i} L^i\right) X(n) = \left(1 + \sum_{k=1}^{q} \theta_{q,k} L^k\right) e(n), \tag{2.53}$$

gde je sa L označen operator kašnjenja.

Generalizacijom ARMA modela, dobijaju se ARIMA (*autoregressive integrated moving average*) procesi,

$$\left(1 - \sum_{i=1}^{p} \phi_{p,i} L^i\right) (1-L)^d X(n) = \left(1 + \sum_{k=1}^{q} \theta_{q,k} L^k\right) e(n). \tag{2.54}$$

Broj telefonskih poziva tokom zadatog intervala vremena se modelira Poissonovim procesom:

$$\mathrm{P}\left(X(t) = k\right) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \quad k = 0, 1, 2, \ldots \tag{2.55}$$

Zadržimo se na nekim osobinama Poissonovog procesa. Lako se pokazuje da su njegovo očekivanje i varijansa dobijeni usrednjavanjem po vremenu

$$\mathrm{E}X(t) = \mathrm{Var}X(t) = \lambda t, \tag{2.56}$$

dok je autokorelaciona funkcija

$$R_{XX}(t_1, t_2) = \lambda^2 t_1 t_2 + \lambda \min(t_1, t_2), \tag{2.57}$$

pa je ovo primer nestacionarnog procesa, jer mu numeričke karakteristike zavise od trenutka posmatranja.

Vreme između dvaju sukcesivnih Poissonovih događaja ima eksponencijalnu raspodelu,

$$f_W(w) = \lambda e^{-\lambda w}, \quad w > 0. \tag{2.58}$$

Pošto nije stacionaran, Poissonov proces se po pravilu simulira indirektno, tako što se simuliraju vremena međudolazaka uzimanjem odbiraka nezavisnih eksponencijalno raspodeljenih slučajnih promenljivih.

2.7.2 Transformacija slučajnog procesa

Neka proces $X(t)$ dolazi na ulaz linearnog vremenski invarijantnog sistema funkcije prenosa $H(f)$. Označimo slučajni proces koji se dobija na izlazu sistema sa $Y(t)$.

Ako je ulazni proces stacionaran, i izlazni proces će biti stacionaran. Ako je ulazni proces Gaussov, takav će biti i izlazni proces.

Matematičko očekivanje, autokorelaciona funkcija i funkcija spektralne gustine snage izlaznog procesa redom su dati izrazima

$$\mathrm{E}Y(t) = \mathrm{E}X(t)H(0), \tag{2.59}$$

$$R_{YY}(\tau) = R_{XX}(\tau) * h(\tau) * h(-\tau), \tag{2.60}$$

$$S_{YY}(f) = S_{XX}(f)\left|H(f)\right|^2, \tag{2.61}$$

gde je $h(\tau)$ impulsni odziv posmatranoga linearnog sistema.

2.8 Generisanje slučajnih brojeva

Nakon što smo videli u čemu je značaj slučajnih procesa u simulaciji i koji su matematički modeli nekih važnih procesa, pogledajmo i kako se u praksi generišu slučajni brojevi koji im odgovaraju.

2.8.1 Generisanje uniformno raspodeljene slučajne promenljive

Jednostavna i popularna metoda za generisanje uniformno raspodeljenih slučajnih brojeva je tzv. linearna kongruentna metoda, koja je opisana izrazom

$$X(k) \equiv [aX(k-1) + c] \pmod{M}. \tag{2.62}$$

Sekvenca X dobijena na ovaj način uzima celobrojne vrednosti iz opsega $[0, M-1]$. U prethodnom izrazu, $M > 0$ je veliki broj koji se naziva modulom i koji predstavlja maksimalan mogući period generisane sekvence. Vrednost parametra a je u opsegu $(0, M)$, c je tzv. inkrement (0 ili 1), dok se početna vrednost $X(0) \in (0, M)$ naziva *seed*.

Uobičajene vrednosti parametara na tridesetdvobitnim procesorima su

- $a = 16\,807$, $c = 0$, $M = 2^{31} - 1$, što daje sekvencu perioda $T = 2^{31} - 2$ ili
- $a = 69\,069$, $c = 1$, $M = 2^{32}$, što daje sekvencu perioda $T = 2^{32}$.

Dobijena celobrojna sekvenca $X(k)$ može se transformisati na interval relnih brojeva $[0, 1)$ na sledeći način:

$$U(k) = \frac{X(k)}{M}. \tag{2.63}$$

Poznati primer lošeg kongruentnog generatora je nekada široko korišćeni RANDU, s parametrima $a = 65\,539$, $c = 0$ i $M = 2^{31}$.

2.8.2 Generisanje slučajnih brojeva iz proizvoljne raspodele

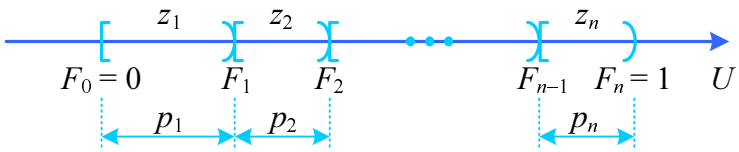
Opisaćemo transformacionu metodu za generisanje slučajnih brojeva.

Neka je Z kontinualna slučajna promenljiva s funkcijom gustine verovatnoće $f_Z(z)$ koju treba generisati. Može se pokazati da je slučajna promenljiva $U = F_Z(z)$, gde je $F_Z(z)$ funkcija raspodele (CDF) za Z, uniformno raspodeljena na intervalu $[0, 1]$. To znači da traženu slučajnu promenljivu Z možemo generisati tako što ćemo prvo generisati $U \sim \text{Unif}\,[0, 1]$ i potom primeniti transformaciju

$$Z = F_Z^{-1}(U), \tag{2.64}$$

u slučaju kad inverzija postoji.

Neka sada treba generisati diskretnu slučajnu promenljivu Z, za koju važi $\mathrm{P}(Z = z_i) = p_i$ i $F_i = \sum_{j=1}^{i} p_j$. Pretpostavićemo da postoji konačno mnogo ishoda z_i, što je najčešći slučaj u primenama. Generisaćemo kontinualnu slučajnu promenljivu U, uniformno raspodeljenu na intervalu $[0, 1)$ (ili $[0, 1]$). Prema slici 2.4, ako je $F_{i-1} \leqslant U < F_i$, generisani slučajni broj biće z_i.



Slika 2.4: Genersanje diskretnih slučajnih brojeva transformacionom metodom.

Pogledajmo sada kako ćemo ove ideje primeniti za generisanje nekih konkretnih raspodela.

Generisanje eksponencijalno raspodeljene slučajne promenljive

Za eksponencijalnu raspodelu je $f_Z(z) = \lambda\, e^{-\lambda z}$ i $F_Z(z) = 1 - e^{-\lambda z}$. Stavimo li $U = F_Z(z)$, imaćemo

$$Z = -\frac{1}{\lambda} \ln(1 - U). \tag{2.65}$$

Ako je $U \sim \mathrm{Unif}[0,1]$, onda je i $1-U \sim \mathrm{Unif}[0,1]$, pa će gornji rezultat statistički biti ekvivalentan sa

$$Z = -\frac{1}{\lambda} \ln U. \tag{2.66}$$

Generisanje geometrijski raspodeljene slučajne promenljive

Funkcija raspodele verovatnoće slučajne promenljive koja ima geometrijsku raspodelu je

$$\mathrm{P}(X = k) = p(1-p)^{k-1}, \quad k = 1, 2, \ldots \tag{2.67}$$

Neka je Z eksponencijalno raspodeljena slučajna promenljiva. Za nju važi

$$\begin{aligned} \mathrm{P}(n < Z \leqslant n+1) &= 1 - e^{-\lambda(n+1)} - \left(1 - e^{-\lambda n}\right) = \\ &= e^{-\lambda n}\left(1 - e^{-\lambda}\right). \end{aligned} \tag{2.68}$$

Ako formalno uvedemo da je $p = 1 - e^{-\lambda}$ verovatnoća uspeha u pojedinačnom Bernoullijevom eksperimentu, odavde ćemo imati

$$\mathrm{P}(n < Z \leqslant n+1) = (1-p)^n p,$$

što odgovara verovatnoći $\mathrm{P}(X = n+1)$, gde je X geometrijski raspodeljena slučajna promenljiva.

U operativnom smislu, prvo ćemo generisati $U \sim \mathrm{Unif}[0,1]$. Potom ćemo primeniti transformaciju

$$Z = \frac{\ln U}{\ln(1-p)}, \tag{2.69}$$

čime se dobija $Z \sim \mathrm{Exp}\left(-\ln(1-p)\right)$. Da bismo dobili traženu geometrijski raspodeljenu slučajnu promenljivu, izvršićemo još jednu transformaciju:

$$X = [1 + Z]. \tag{2.70}$$

Generisanje Erlangove raspodele

Slučajni broj iz raspodele $\gamma(\alpha, \beta)$ u kojoj je $\alpha \in \mathbb{N}$, generiše se po sledećem algoritmu:

1. $X := 0$
2. Generiše se $V \sim \mathrm{Exp}(1)$.
3. $X := X + V$
4. Ako je $\alpha = 1$:
   $X := \beta X$
   izlaz algoritma je X

   u suprotnom:
   $\alpha := \alpha - 1$
   vrati se na korak 2.

Generisanje Poissonove slučajne promenljive

Sledeći algoritam služi za generisanje slučajnih brojeva iz raspodele Poiss(λ):

1. $A := 1$, $k := 0$
2. Generiše se $U \sim \text{Unif}\,[0,1]$.
3. $A := AU$
4. Ako je $A < e^{\lambda}$:
   $X := \beta X$
   izlaz algoritma je $X = k$
   u suprotnom:
   $k := k + 1$
   vrati se na korak 2.

Generisanje Gaussove slučajne promenljive

Ne postoji analitički oblik inverzne funkcije Gaussove raspodele, pa se stoga ovde ne može primeniti transformaciona metoda. U nastavku ćemo opisati polarnu metodu i metodu sume 12, koje se najčešće koriste.

Polarnom metodom se generiše par nezavisnih Gaussovih slučajnih brojeva. Neka su U_1 i U_2 nezavisne slučajne promenljive iz raspodele Unif $[0,1]$. Formirajmo nove slučajne promenljive

$$R = \sqrt{\ln \frac{1}{U_1^2}}, \tag{2.71}$$

$$\Theta = 2\pi U_2, \tag{2.72}$$

koje su očigledno nezavisne. Može se pokazati da slučajne promenljive

$$Z_1 = R\cos\Theta \tag{2.73}$$

i

$$Z_2 = R\sin\Theta \tag{2.74}$$

predstavljaju par nezavisnih slučajnih promenljivih iz standardne normalne raspodele.

U literaturi su opisane implementacije polarnog metoda koje su računski efikasnije od ove osnovne ideje. Prema jednoj od njih, izbor U_1 i U_2 se ponavlja sve dok se njihove vrednosti ne nađu unutar kruga jediničnog poluprečnika, tj. dok ne bude važilo

$$\rho = U_1^2 + U_2^2 < 1, \tag{2.75}$$

a potom se primeni transformacija

$$Z_1 = U_1\sqrt{\frac{-2\ln\rho}{\rho}}, \tag{2.76}$$

$$Z_2 = U_2\sqrt{\frac{-2\ln\rho}{\rho}}. \tag{2.77}$$

Metoda sume 12 se zasniva na centralnoj graničnoj teoremi. Po ovoj metodi, prvo se generiše 12 nezavisnih slučajnih promenljivih U_i, koje su uniformno raspodeljene na intervalu $[0,1]$. Potom se formira nova slučajna promenljiva

$$Y = \sum_{i=1}^{12} U_i - 6, \tag{2.78}$$

koja veoma približno predstavlja uzorak standardne normalne raspodele na intervalu $[-6,6]$.

Osvrnimo se na razloge za izbor parametara u ovoj metodi. Prema CGT, zbir iid slučajnih promenljivih iz bilo koje raspodele konvergira ka Gaussovoj. U primenama bi bilo poželjno da polazna raspodela bude jednostavna za generisanje, kao i da potreban broj sabiraka ne bude prevelik. Raspodela $\mathrm{Unif}\,[0,1]$ je izabrana zato što predstavlja osnovu za izvođenje drugih raspodela, pa su razvijeni efikasni algoritmi za njenu računarsku implementaciju. Matematičko očekivanje ove raspodele je $1/2$, a varijansa $1/12$. Uzmemo li dvanaest sabiraka, dobiće se zbir jedinične varijanse, pa otpada potreba za operacijom deljenja u (2.46). Matematičko očekivanje ovakvog zbira biće 6, pa oduzimanjem ove vrednosti dobijamo $\mathrm{E}Y = 0$. U odeljku 6.6.3, videćemo da se ovako dobijeni slučajni brojevi zaista dobro slažu s normalnom raspodelom.

Uskopojasni Gaussov šum jednostrane spektralne gustine srednje snage $N_0/2$, koji zauzima opseg frekvencija $|f| < f_s/2$ generiše se tako što se generiše slučajan broj iz standardne normalne raspodele i pomnoži faktorom $N_0 f_s/2$.

Generisanje Gaussovog slučajnog vektora

Neka su Z_1 i Z_2 nezavisne slučajne promenljive iz standardne normalne raspodele i neka je a jedno rešenje jednačine

$$\frac{\rho^2}{4a^2} + a^2 = 1. \tag{2.79}$$

Slučajne promenljive

$$X_1 = a\sigma_1 Z_1 + \frac{\rho\sigma_1}{2a} Z_2 \tag{2.80}$$

i

$$X_2 = \frac{\rho\sigma_2}{2a} Z_1 + a\sigma_2 Z_2 \tag{2.81}$$

odgovaraju normalnom dvodimenzionom slučajnom vektoru s koeficijentom korelacije ρ i vektorom varijanse (σ_1, σ_2).

U više od dve dimenzije, mormalni vektor sa zadatom kovarijacionom matricom $\mathbf{C}$ generiše se iz vektora nezavisnih standardnih normalnih slučajnih promenljivih $\mathbf{Z}$ po obrascu

$$\mathbf{X} = \mathbf{C}^{1/2}\,\mathbf{Z}. \tag{2.82}$$

Generisanje slučajnih digitalnih sekvenci

Princip sa slike 2.4 možemo direktno primeniti za generisanje slučajnih digitalnih sekvenci.

Binarnu sekvencu X u kojoj je verovatnoća pojavljivanja bita „1" p generišemo tako što generišemo slučajni broj U, uniformno raspodeljen na $[0, 1]$ i potom primenimo pravilo odlučivanja:

$$X = \begin{cases} 1, & 0 \leqslant U < p \\ 0, & \text{u suprotnom} \end{cases}. \tag{2.83}$$

Ekvivalentno postupamo i pri generisanju M-arnih sekvenci, s tim što tada postavljamo $M - 1$ prag odlučivanja.

2.8.3 Testiranje generatora slučajnih brojeva

Tri su važna aspekta generatora slučajnih brojeva koji se primenjuju u telekomunikacijama: algebarske (vremenske) odlike, statističke odlike i računska složenost algoritma. Algebarske odlike se odnose na strukturu i period generisane sekvence, a statističke na raspodelu njenih vrednosti. Imperativ je da se na što jednostavniji način generiše slučajna sekvenca koja će biti stacionarna, nekorelisana i maksimalnog perioda, dužeg od trajanja simulacije.

Stacionarnost sekvence skučajnih brojeva se proverava tako što se dugačka sekvenca izdeli na nepreklapajuće segmente, za koje se izračunaju momenti; za stacionarne sekvence ne bi trebalo dobiti značajnije razlike u njihovim vrednostima.

Kao test korelisanosti može poslužiti Durbin-Watsonov test:

$$D = \frac{\sum_{n=2}^{N} [X(n) - X(n-1)]^2}{\sum_{n=1}^{N} X^2(n)}. \tag{2.84}$$

Ako su $X(n)$ i $X(n-1)$ nekorelisani, dobiće se vrednost $D = 2$. Rezultat $D \ll 2$ ukazuje na jaku pozitivnu korelisanost između elemenata sekvence, dok rezultat $D \approx 4$ ukazuje na jaku negativnu korelisanost. Slični testovi postoje i za ispitivanje korelisanosti više

od dva sukcesivna elementa sekvence. Pomenuti kongruentni generator RANDU se masovno koristio sve dok se nije ustanovilo da ne prolazi test korelisanosti.

Pripadnost generisanih brojeva pretpostavljenoj raspodeli se testira nekim od statističkih testova s kojima ćemo se upoznati nešto kasnije, u odeljku 6.6. Sada ćemo razmotriti tzv. test serija, koji pomaže da se utvrdi koliko je zadati binarni niz zaista slučajan, što će ujedno biti i dobra prilika da se podsetimo kombinatorike.

Odredimo verovatnoću da se u slučajnom nizu od m nula i n jedinica javi r serija jedinica, pri čemu se pod serijom podrazumeva uzastopno pojavljivanje istog znaka, 0 ili 1.

Ako niz ima r serija jednica, između njih će sigurno biti $r-1$ serija nula. Na početku i na kraju niza se mogu javiti još do dve serije nula. Označimo li sa y_j broj nula u j-toj seriji, možemo pisati

$$y_1 + y_2 + \cdots + y_{r+1} = m, \tag{2.85}$$

gde je $y_1, y_{r+1} \geqslant 0$ i $y_i > 0$, $i = 1, 2, \ldots, r$. Dodamo li obema stranama jednačine 2, dobićemo

$$(y_1 + 1) + y_2 + \cdots + (y_{r+1} + 1) = m + 2. \tag{2.86}$$

Prethodnu jednačinu možemo napisati kao

$$\bar{y}_1 + \bar{y}_2 + \cdots + \bar{y}_{r+1} = m + 2\,, \quad \bar{y}_i > 0. \tag{2.87}$$

Interesuje nas broj rešenja ove jednačine u skupu prirodnih brojeva. Do odgovora na to pitanje možemo doći rezonovanjem koje je ilustrovano na slici.

Slika 2.5: Kombinatorna interpretacija jednačine (2.87).

Neka postoje $m+2$ kružića koje treba rastaviti crticama. Od $m+1$ mesta, r mesta za crtice se može izabrati na $\binom{m+1}{r}$ načina, što odgovara broju mogućnosti za formiranje serije nula.

Rasporedili smo nule, sada treba da rasporedimo jedinice i to njih n u r serija. Treba da odredimo broj rešenja jednačine

$$x_1 + x_2 + \cdots + x_r = n \tag{2.88}$$

u skupu $\mathbb{N}$. Primenom istog rezona kao za nule, dobijamo da se jedinice mogu rasporediti na $\binom{n-1}{r-1}$ načina. Ukupan broj mogućnosti za formiranje binarnog niza s m nula i n jedinica, koji ima r serija jedinica stoga je $\binom{m+1}{r}\binom{n-1}{r-1}$.

S druge strane, ukupan broj nizova s m nula i n jedinica je $\binom{m+n}{m} = \binom{m+n}{n}$. Tražena verovatnoća stoga je

$$\mathrm{P} = \frac{\binom{m+1}{r}\binom{n-1}{r-1}}{\binom{m+n}{n}}. \tag{2.89}$$

Test se sprovodi tako što se u zadatoj sekvenci odrede vrednosti m, n i r. Potom se izračuna vrednost izraza (2.89) i uporedi sa zadatim pragom (tipično 5%). Ako je P manje od praga, smatra se da je sekvenca prošla test.

Kao primer, izračunali smo vrednosti verovatnoće P za nekoliko binarnih sekvenci za koje je $m = n = 4$ i rezultate prikazali u tabeli 2.2.

Tabela 2.2: Primeri testa serija.

Sekvenca	r	P
11110000	1	0,0714
11001100	2	0,4286
11001010	3	0,4286
10101010	4	0,0714

2.8.4 Generisanje slučajnih brojeva pomoću računara

Pregled naredbi za generisanje slučajnih brojeva iz nekih raspodela u programima GNU Octave i Python 2.7 (moduo `random`) dat je u tabeli 2.3.

Tabela 2.3: Naredbe za generisanje slučajnih brojeva.

Raspodela	GNU Octave	Python
		`from random import *`
Diskretna uniformna	`unidrnd`	`randint`
Binomna	`binornd`	
Negativna binomna	`nbinrnd`	
Geometrijska	`geornd`	
Poissonova	`poissrnd`	
Kontinualna uniformna	`unifrnd`	`uniform`
Normalna	`normrnd`	`gauss` ili `normalvariate`
Eksponencijalna	`exprnd`	`expovariate`
Gama	`gamrnd`	`gammavariate`

Daleko veći broj raspodela, preko 90, dostupan je u Pythonovom SciPy modulu `stats`, korišćenjem metoda `rvs` (*random variates*). Evo kako se na ovaj način mogu generisati neki slučajni brojevi:

```
from scipy import stats
stats.bernoulli.rvs  # Bernoullijeva raspodela
stats.binom.rvs      # binomna raspodela
```

```
stats.nbinom.rvs     # negativna binomna raspodela
stats.geom.rvs       # geometrijska raspodela
stats.poisson.rvs    # poissonova raspodela
stats.norm.rvs       # normalna raspodela
stats.erlang.rvs     # Erlangova raspodela
stats.expon.rvs      # eksponencijalna raspodela
stats.rayleigh.rvs   # Rayleighjeva raspodela
```

Poglavlje 3

Metodologija simulacije

U ovome poglavlju ćemo razmotriti metodološke postavke simulacije. Upoznaćemo se s različitim vrstama simulacije, videćemo koje su odlike simulacionog softvera i daćemo savete za pisanje simulacionih programa.

3.1 Simulacija diskretnih događaja

Događaj predstavlja svaku pojavu koja izaziva promenu stanja sistema. Prema mestu nastanka, događaji mogu biti eksterni i interni. Eksterni događaji ne zavise od modela i rezultat su uticaja okoline na sistem; primer eksternih događaja su dolasci korisničkih paketa u čvor mreže. S druge strane, interni događaji, kao što je npr. opsluživanje paketa, zavise od modela i generišu se u njemu.

U najvećem broju slučajeva, telekomunikacioni sistem se modelira tako da se događaji dešavaju u prebrojivo mnogo trenutaka vremena; ovakav vid simulacije naziva se simulacijom diskretnih događaja ili, kraće, diskretnom simulacijom. U nastavku ovoga odeljka, razmotrićemo njene najznačajnije odlike.

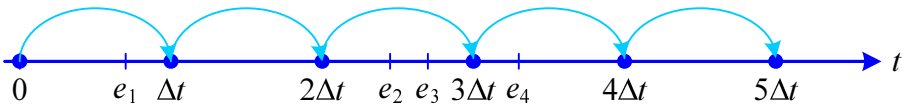
3.1.1 Mehanizmi pomaka vremena

Bitan element simulacije diskretnih događaja je simulacioni sat. To je promenljiva koja sadrži trenutnu vrednost simulacionog vremena, u kome je potpuno poznato stanje sistema u simulacionom okruženju. Simulacioni sat se inicijalizuje proizvoljno, najčešće na nultu vrednost i raste tokom progresije stanja sistema[1].

[1] Izuzetak su tzv. rollback procedure u paralelnoj simulaciji, koje dovode do vraćanja simulacionog vremena i koje ćemo objasniti na strani 35.

Postoje dva načina za pomak simulacionog vremena i to pomak za konstantni priraštaj i pomak na naredni događaj.

Mehanizam pomaka vremena za konstantni priraštaj ilustrovan je na slici 3.1. Simulacioni sat ovde neprekidno napreduje za konstantni interval Δt.



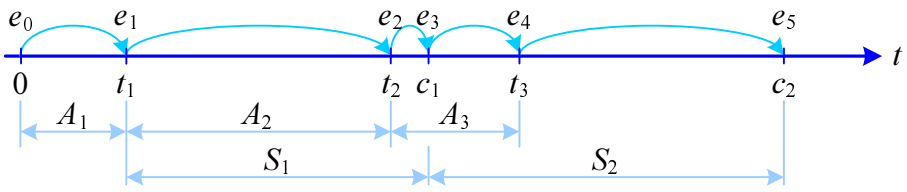
Slika 3.1: Pomak vremena za konstantni priraštaj.

Nakon ažuriranja sata, proverava se da li je tokom prethodnog intervala bilo događaja, koji su na slici označeni slovom e. Ako je odgovor potvrdan, smatra se da su se događaji desili na kraju prethodnog intervala, pa se u skladu s tim ažuriraju stanje sistema i statistički brojači.

Dobra strana ovoga pristupa je jednostavnost implementacije; mane su pojava praznog hoda, koji odgovara obradi intervala bez događaja (drugi i peti interval na slici), kao i to što se unosi greška zbog pomeranja događaja na kraj intervala. Ovaj problem je naročito izražen u slučaju kada se tokom trajanja jednog intervala desi više događaja (treći interval na slici), jer se postavlja pitanje kojim redom ih treba obraditi. Moguće rešenje ovih problema je u skraćivanju intervala Δt, ali tada se povećava količina podataka za obradu i usporava se simulacija.

Mehanizam pomaka vremena na naredni događaj podrazumeva da se simulacioni sat inicijalizuje na početnu vrednost (npr. 0), da se odrede vremena nastupanja budućih događaja i da se potom sat pomera na vreme kada će nastupiti prvi od njih. Nakon pomaka simulacionog sata, obrađuje se događaj, što znači da se ažuriraju stanje sistema, statistički brojači i vremena nastupanja budućih događaja. Na ovaj način se preskaču periodi neaktivnosti, u kojima nema događaja, pa se ovaj pristup po pravilu primenjuje u praksi.

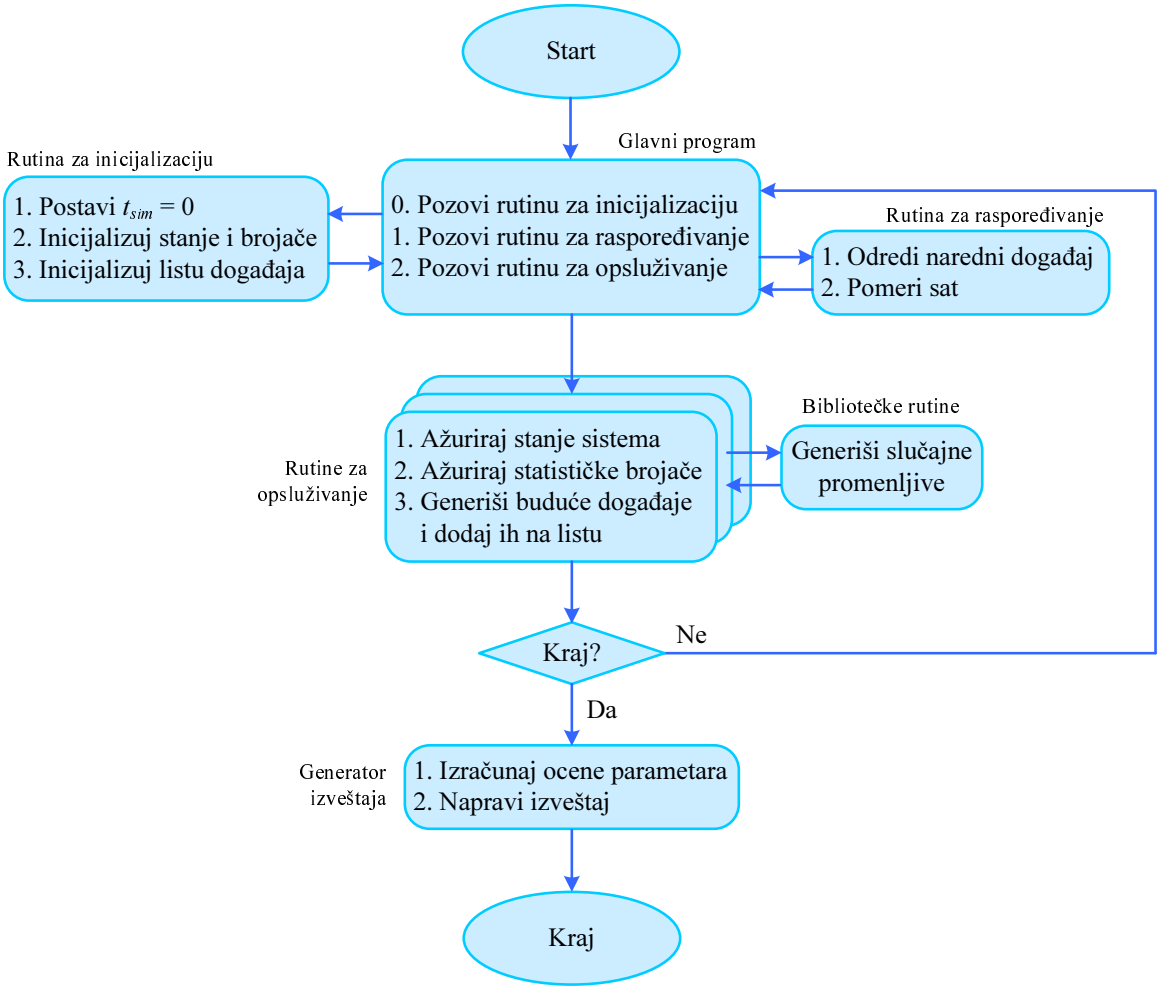
Pomak vremena na naredni događaj ilustrovan je na slici 3.2 za primer servisnog sistema. Sa t_i označen je trenutak dolaska i-tog korisnika u sistem; $A_i = t_i - t_{i-1}$ je vreme između dolazaka korisnika $i-1$ i i; S_i je trajanje obrade i-tog korisnika, c_i trenutak njegovog izlaska iz sistema, dok je e_j j-ti događaj, bilo koje prirode.



Slika 3.2: Pomak vremena na naredni događaj.

3.1.2 Organizacija simulacije diskretnih događaja

Simulacija diskretnih događaja bi se, u principu, mogla izvesti i ručno, korak po korak, ali je takav pristup besmislen u praksi, zbog velike količine podataka koju treba obraditi. Organizacija „školski" napisanog računarskog programa za simulaciju diskretnih događaja prikazana je na slici.



Slika 3.3: Organizacija simulacije diskretnih događaja.

Tokom trajanja simulacije, održavaju se podaci o simulacionom satu, stanju sistema i listi budućih događaja, koja sadrži vremena njihovog nastupanja. Program je modularno organizovan, pa glavni program komunicira s nizom potprograma (rutina).

Rutina za inicijalizaciju inicijalizuje simulacioni model u početnom trenutku – postavlja vrednosti simulacionog vremena, stanja sistema, statističkih brojača i liste događaja.

Rutina za raspoređivanje događaja određuje koji će se događaj sa liste prvi desiti i potom pomera simulacioni sat ka tome trenutku.

Rutine za opsluživanje događaja su potprogrami koji obrađuju događaje, u smislu da

ažuriraju stanje sistema, statističkih brojača i liste događaja svaki put kada se desi određeni događaj. Za svaku vrstu događaja, u opštem slučaju postoji posebna rutina.

Bibliotečke rutine su matematički potprogrami koji generišu slučajne promenljive sa zadatom raspodelom.

Generator izveštaja je programski modul koji po završetku simulacije računa ocene parmetara performansi sistema na osnovu vrednosti statističkih brojača, pravi izveštaj i prezentuje ga korisniku.

Glavni program prilikom pokretanja pozove rutinu za inicijalizaciju, a potom ciklično poziva rutinu za raspoređivanje događaja i rutinu za njihovo opsluživanje. Glavni program potom proverava ispunjenost uslova za završetak simulacije, u kome slučaju poziva generator izveštaja i okončava simulaciju.

3.2 Kontinualna simulacija

Kontinualna simulacija podrazumeva da se promenljive stanja kontinualno menjaju u vremenu. Veliki broj realnih sistema je (dominantno) kontinualne prirode, što znači da su opisani integrodiferencijalnim jednačinama. Za analizu ovakvih sistema, na raspolaganju su sledeće mogućnosti:

- sistem možemo analitički rešiti,
- sistem možemo emulirati na analognom računaru, ili
- sistem možemo simulirati na digitalnom računaru.

Analitičko rešenje je jedino koje tačno odgovara usvojenom modelu, ali do njega često nije lako doći, ili ne postoji u zatvorenoj formi. Analogni računari su postali stvar istorije, pa tako preostaje simulacija sistema na digitalnom računaru, što podrazumeva diskretizaciju modela. Probleme koji se pri tome mogu javiti razmotrićemo u narednom poglavlju.

U praksi se sreće i kombinovana diskretno-kontinualna simulacija. Ona se primenjuje u sledećim situacijama:

- kada diskretni događaj izaziva diskretnu promenu vrednosti kontinualne promenljive stanja,
- kada diskretan događaj menja funkcionalnu vezu kontinualnih promenljivih stanja, ili
- kada kontinualna promenljiva stanja dostigne prag vrednosti koji potom izaziva neki diskretan događaj.

Ako diskretan događaj zamislimo kao otvaranje ili zatvaranje prekidača, lako ćemo zaključiti da su ove situacije široko zastupljene u električnim kolima.

3.3 Tabelarna simulacija

Za simuliranje jednostavnijih modela mogu biti dovoljni i programi za tabelarna izračunavanja, kao što je LibreOffice Calc ili, ranije, OpenOffice Calc, koji nude funkcionalnosti generisanja slučajnih brojeva, osnovnih matematičkih proračunavanja, statističke obrade i grafičkog predstavljanja rezultata. Ovakav vid simulacije se naziva tabelarnom ili „spreadsheet" simulacijom.

3.4 Hibridna simulacija

Hibridna simulacija, koja se zove i „hardware in the loop", podrazumeva korišćenje hardverskih sklopova kao elemenata simulacionog okruženja. Ovi sklopovi mogu da predstavljaju bilo stvarne komponente fizičkog sistema, poput npr. senzora ili aktuatora nekog postrojenja, bilo njihove hardverske emulacije.

Uključivanje hardvera u simulaciono okruženje je motivisano željom za što većom vernošću simulacionog modela. Najveći problem u praktičnim implementacijama leži upravo u sprezanju hardverskih modula sa softverom, što otvara pitanja poput A/D i D/A konverzije, usklađivanja brzina rada i sl. Stoga se po pravilu ovakvom pristupu pribegava onda kada je matematičko modeliranje nekih blokova isuviše komplikovano ili iz drugih razloga neprimenljivo.

3.5 Paralelna simulacija

Do sada smo razmatrali sekvencijalni pristup simulaciji, po kome je simulacioni model jedinstvena celina koja se izvršava na jednom procesoru. Da bi se ubrzalo izvršavanje simulacije, ili da bi se omogućilo simuliranje složenijih modela, često se pribegava paralelnoj simulaciji, gde se različiti segmenti jednog simulacionog modela izvršavaju na procesorima koji rade paralelno. Zanimljivo je primetiti da su višeprocesorski sistemi nekada bili superračunari, a danas su to standardni PC.

Paralelna simulacija ima dve varijante, vremensku i prostornu.

3.5.1 Vremenski paralelna simulacija

Algoritmi vremenski paralelne simulacije još uvek nisu dovoljno opšti, već se odnose na rešavanje pojedinih klasa problema. Zajedničko za njih je da se osa simulacionog vremena deli na nepreklapajuće intervale, koji se dodeljuju različitim procesorima. Svaki procesor pretpostavlja početne uslove za interval čije mu je simuliranje dodeljeno i

potom obavlja simulaciju; po njenom završetku, započinje usklađivanje pretpostavljenih početnih uslova s dobijenim izlazima drugih procesora. Ovaj tzv. fix-up proces u opštem slučaju može biti netrivijalan, pa će se simulacije na nekim intervalima, a u najgorem slučaju na svim sem na prvom, morati ponoviti.

Vremenski paralelna simulacija daje dobre rezultate npr. kod analize statističkih multipleksera.

3.5.2 Prostorno paralelna simulacija

U ovom pristupu, simulacioni model se dekomponuje na logičke procese, koji se izvršavaju na zasebnim procesorima. Svaki logički proces je diskretan simulacioni model za sebe, pa procesori na ovaj način obrađuju jedan segment „velikog" modela. Na primer, u simulaciji telekomunikacionih mreža, velika mreža bi se mogla izdeliti na podmreže, od kojih bi svaka predstavljala jedan logički proces; iskustvo pokazuje da je podelu na podmreže poželjno izvršiti tako da se preseku linkovi malog kapaciteta i velikog propagacionog kašnjenja, kao i da se procesori što ravnomernije opterete.

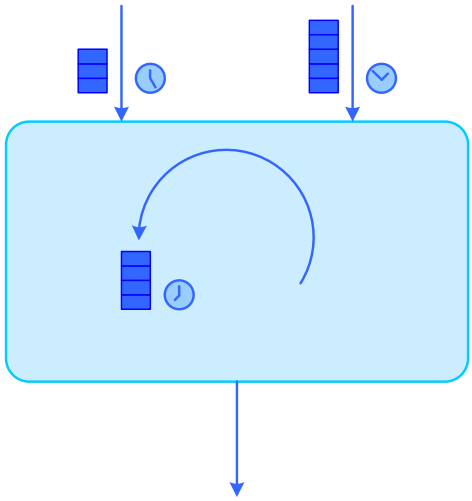
Logički procesi uzajamno komuniciraju preko poruka s vremenskim oznakama. Da bi prostorno paralelna simulacija dala isti rezultat kao i sekvencijalna, neophodno je ostvariti sinhronizaciju logičkih procesa. U operativnom smislu, to znači da logičke procese treba obrađivati po rastućem vremenskom redosledu, što je poznato kao uslov lokalne kauzalnosti. U literaturi su opisane dve grupe algoritama za sinhronizaciju logičkih procesa, konzervativni i optimistični.

Konzervativni algoritmi

Konzervativni algoritmi ne dozvoljavaju narušavanje uslova lokalne kauzalnosti tako što logički procesi čekaju da i drugi napreduju do njihovog simulacionog vremena, odnosno da obrade događaj s najmanjom vremenskom oznakom.

Konzervativni algoritmi prve generacije koriste tzv. null poruke. Pretpostavlja se da se ne menjaju veze između logičkih procesa, tj. da je poznato koji logički procesi uzajamno razmenjuju poruke. Na svakom dolaznom linku u logički proces postoji FIFO red za poruke i sat koji pokazuje vremensku oznaku prve poruke u redu ukoliko red nije prazan, ili poslednje primljene poruke u suprotnom. Identičan red sa satom postoji i u samom logičkom procesu i u njega se smeštaju poruke koje logički proces generiše za samog sebe. Ovo je ilustrovano na slici 3.4.

Logički proces bira red čiji sat pokazuje najmanje vreme. Ukoliko ovaj red nije prazan, logički proces uzima poruku iz njega i obrađuje je; u suprotnom, ako je red prazan, logički proces se blokira i čeka ili da u prazan red stigne nova poruka, ili da simulaciono vreme napreduje ka vremenu sata u nekom od redova s porukama. Ne treba posebno



Slika 3.4: Konzervativna sinhronizacija logičkih procesa.

naglašavati da ovo čekanje degradira prednosti paralelizma; ono što nije vidljivo na prvi pogled je da se logički proces nikada neće blokirati zbog reda u kome su poruke koje je namenio samom sebi.

Ciklus praznih redova čekanja čiji satovi pokazuju mala vremena naziva se „deadlock" i može značajno usporiti izvršavanje simulacije. Jedno rešenje za izbegavanje deadlocka su tzv. null poruke, koje ne donose posao logičkom procesu, već ga svojom vremenskom oznakom T_{null} obaveštavaju da proces koji ih je poslao neće kasnije poslati poruku s oznakom manjom od T_{null}. Iako null poruke skraćuju periode čekanja, nije dobro da se često šalju (što je npr. slučaj kada su logički procesi uzajamno dobro povezani), jer mogu zagušiti logičke procese.

Konzervativni algoritmi druge generacije izbegavaju null poruke tako što ili detektuju „deadlock" i razbijaju ga, ili procenjuju donju granicu budućih vremenskih oznaka.

Optimistični algoritmi

Optimistični algoritmi sinhronizacije dozvoljavaju narušavanja uslova lokalne kauzalnosti, ali ih otkrivaju i potom pokreću proceduru oporavka od njih. Optimistični algoritmi omogućavaju veći stepen paralelizma i transparentniji su za aplikacije, pa je programiranje jednostavnije; mana im je to što zbog velikog broja ponovljenih izračunavanja mogu imati lošije performanse nego konzervativni algoritmi.

Najpoznatiji optimistični algoritam je „time warp". Kada detektuje narušavanje uslova lokalne kauzalnosti, ovaj algoritam pokreće proceduru koja je poznata kao rollback: simulacioni sat se premotava unazad, a drugim logičkim procesima, kojima su u međuvremenu poslate poruke, sada se šalju antiporuke. Ako se u redu čekanja nađu poruka i njena antiporuka, uzajamno će se poništiti i neće biti drugih posledica po odredišni logički proces. U suprotnom, ako je odredišni logički proces već obradio poruku, prijem

antiporuke će i u njemu inicirati rollback.

Primetimo da neke operacije, kao što su npr. ulaz i izlaz, nije mogiće poništiti. Veliki broj oporavaka znači i ponovljena izračunavanja, čime se gubi vreme. Zbog rollbacka je, u principu, potrebno i više memorije, jer treba čuvati istoriju stanja logičkih procesa. Zbog svega ovoga, u implementacijama ne treba dozvoliti da neki logički proces previše odmakne ispred ostalih, jer bi njegov rollback najverovatnije izazvao krupne poremećaje u izvršavanju simulacije.

3.6 Distribuirana simulacija

Distribuirana simulacija podrazumeva da je „veliki" simulacioni model kompozicija manjih, koji se izvršavaju na umreženim računarima. Standardom IEEE 1516 opisana je arhitektura distribuirane simulacije poznata kao HLA (*High Level Architecture*), koju je razvila Kancelarija za modeliranje i simulaciju pri Ministarstvu odbrane SAD.

Čest pristup u inženjerskoj praksi je da se distribuirana simulacija implementira tako što se jednaki simulacioni modeli istovremeno izvršavaju na nekoliko računara, da bi se u istom fizičkom vremenu ostvarilo što više replikacija.

3.7 Simulacioni softver

Simulacija se može programirati u programskim jezicima opšte namene, ili u specijalizovanom simulacionom softveru, koji može biti opštenamenski ili aplikaciono orijentisan. Čest izbor za pisanje simulacionih programa su programski jezik C i objektno-orijentisan jezik Python, koji ima i poseban modul za podršku simulacije, SimPy. Najveći broj primera u ovoj knjizi napisan je u matematičkom programskom paketu opšte namene GNU Octave 3.0.3 i programskom jeziku Python 2.7.2; oba spadaju u kategoriju slobodnog softvera. Poznati opštenamenski simulacioni softver je GPSS/H, dok je primer aplikaciono orijentisanog softvera *ns*-3, koji se koristi za simulaciju telekomunikacionih mreža.

Odlike simulacionog softvera možemo svrstati u nekoliko kategorija. Opšte odlike su fleksibilnost u formiranju modela, grafički korisnički interfejs, funkcionalnost debagovanja, brzina izvršavanja, mogućnost sprezanja s drugim programima, uslovi korišćenja i distribuiranja i cena. Narednu kategoriju čine hardverski i softverski zahtevi za platformu na kojoj se taj softver izvršava. U svim primenama su bitne statističke funkcionalnosti koje softver nudi, vrsta izlaznog izveštaja i podrška grafičkom prikazu rezultata, dok je ponegde bitna i mogućnost animacije. Na kraju, treba voditi računa i o podršci korisnicima, u vidu adekvatne dokumentacije, postojanja udžbenika, demo verzije programa itd.

Na kraju ovoga odeljka, daćemo nekoliko „školskih" saveta za pisanje dobrih simulacionih programa.

Pre svega, svaki složeniji program treba pisati u vidu modula, tj. potprograma. Na taj način, programski kod će biti pregledniji, lakše će se uočiti eventualne greške i nedostaci, a olakšaće se i modifikovanje programa u budućnosti.

Neophodno je dokumentovati sve korake u razvoju modela i pisanju programa. Po završetku ove faze rada, programski kod i dokumentaciju treba dati nekome ko nije bio uključen u njihovo pisanje, da ih verifikuje.

Složenu simulaciju prvo treba izvršiti pod pojednostavljenim pretpostavkama, za koje su rezultati unapred poznati, a potom i za različite kombinacije ulaznih parametara i proveriti da li se dobijaju smisleni rezultati. U slučaju problema, posle svakog događaja treba proveravati tzv. trace – stanje sistema, brojače i listu događaja. Čest uzrok nesaglasnosti dobijenih rezultata s očekivanim leži u neadekvatnom generisanju slučajnih brojeva, pa treba proveriti parametre njihovih raspodela.

Kada je to moguće, treba posmatrati animaciju procesa u sistemu.

3.8 Primer

Postavke simulacije diskretnih događaja ćemo ilustrovati na primeru simulacije servisnog sistema M/M/1.

Servisni sistem je matematički model procesa u kome korisnici donose neku količinu posla na obradu. Servisni sistem M/M/1 se sastoji od čekaonice beskonačnog kapaciteta i radionice s jednim serviserom (ili serverom); njegova simbolička oznaka je prikazana na slici 3.5.

Slika 3.5: Servisni sistem M/M/1.

Dolasci korisnika u sistem predstavljaju Poissonov slučajni proces, s prosečnim protokom λ. Količina posla koju korisnici donose je eksponencijalno raspodeljena slučajna promenljiva, s matematičkim očekivanjem $\bar{\tau} = 1/\mu$, gde je μ protok obrade. Smatra se da je populacija korisnika beskonačna i da je disciplina opsluživanja FCFS (*First-Come-First-Served*).

Ukoliko novopridošli korisnik zatiče prazan sistem, on ide pravo do servisera, dok u protivnom u čekaonici čeka da dođe na red.

Zadatak simulacije je da odredi empirijsku zavisnost prosečnog broja korisnika u če-

kaonici, N_Q, prosečnog zadržavanja korisnika u čekaonici (tj. trajanja čekanja), T_Q i prosečnog zadržavanja korisnika u sistemu, T, od iskorišćenosti (opterećenosti) servera, ρ. Analitički poznate zavisnosti su

$$N_Q = \frac{\rho^2}{1-\rho}, \tag{3.1}$$

$$T_Q = \frac{\rho}{\mu - \lambda} \tag{3.2}$$

i

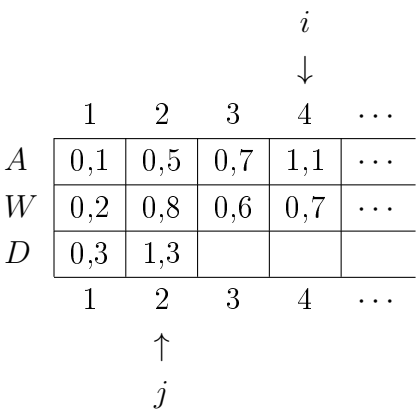
$$T = \frac{1}{\mu - \lambda}, \tag{3.3}$$

pri čemu je $\rho = \lambda/\mu$.

Simulacija je organizovana u vidu simulacije diskretnih događaja, koji odgovaraju dolascima i odlascima korisnika. Zbog jednostavnosti problema i preglednosti rešenja, u nekim delovima je odstupljeno od „školske" organizacije programa sa slike 3.3. Simulacija se izvršava sve dok se ne bude opslužilo k korisnika, tj. uslov za završetak je izlazak k-tog korisnika iz sistema. Radi jednostavnije realizacije programa, pretpostavljeno je da korisnici dolaze i posle k-tog, ali se ne obrađuju. Ukupan broj korisnika koji mogu ući u sistem tokom izvršavanja simulacije stoga je $n > k$.

U simulaciji je iskorišćena osobina Poissonovog procesa da je vreme između dvaju sukcesivnih dolazaka eksponencijalno raspodeljena slučajna promenljiva, s matematičkim očekivanjem $1/\lambda$. Stoga je prvo generisan eksponencijalno raspodeljen niz s vremenima međudolazaka, IA, pa je njihovim sabiranjem dobijen niz s trenucima dolazaka korisnika, A. Niz s količinom posla koji korisnici sobom donose, W, generisan je iz eksponencijalne raspodele. Trenuci odlazaka korisnika iz sistema se smeštaju u niz D.

U programskom kodu se ne vidi eksplicitno simulacioni sat. Njegovu ulogu preuzimaju brojači koji ukazuju na naredni dolazak (i) i naredni odlazak (j). Prema slici 3.6, vrednost simulacionog vremena je $\min(A(i), D(j))$.



Slika 3.6: Nizovi vremena i brojači događaja.

Objasnimo kako se računaju trenuci odlazaka korisnika iz sistema. Nakon odlaska korisnika iz sistema, ispituje se da li je server slobodan. Ako jeste ($i = j$), to znači

da se naredni korisnik (npr. m-ti) neće zadržavati u čekaonici, već će odmah ići na obradu, koju će završiti u trenutku $D(m) = A(m) + W(m)$. Ako server nije slobodan ($i \neq j$), novi korisnik će u čekaonici čekati da dođe na red. U trenutku $D(m-1)$ server se oslobađa i posmatrani korisnik ulazi u radionicu, iz koje će izaći u trenutku $D(m) = D(m-1) + W(m)$.

Obrada dolaska obuhvata samo inkrementiranje brojača obrađenih dolazaka. Nasuprot tome, obrada odlaska obuhvata ažuriranje statističkih brojača, inkrementiranje brojača odlazaka i računanje vremena narednog odlaska.

Za svakog obrađenog korisnika se evidentiraju vremena koja je proveo u čekaonici i u sistemu i smeštaju u nizove t_Q i t, respektivno. Na osnovu ovih podataka, na kraju simulacije, što se dešava u trenutku $t\_stop = D(k)$, računaju se prosečan proj korisnika u čekaonici (N_Q), prosečno zadržavanje korisnika u sistemu (T) i prosečno zadržavanje korisnika u čekaonici (T_Q), na sledeći način:

$$N\_Q = \frac{1}{t\_stop} \sum_{m=1}^{k} t\_Q(m), \tag{3.4}$$

$$T = \frac{1}{k} \sum_{m=1}^{k} t(m), \tag{3.5}$$

$$T\_Q = \frac{1}{k} \sum_{m=1}^{k} t\_Q(m). \tag{3.6}$$

Iskorišćenost servera tokom trajanja simulacije se procenjuje kao

$$rho = \frac{1}{t\_stop} \sum_{m=1}^{k} W(m). \tag{3.7}$$

Programski kodovi u GNU Octave i Pythonu dati su u nastavku. Naglasimo da je pre njihovog izvršavanja potrebno ukloniti dijakritičke simbole iz napomena.

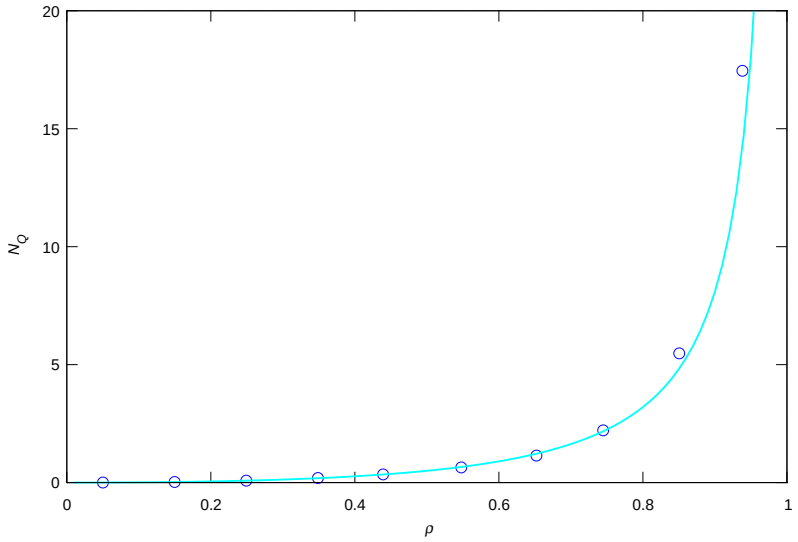
```
function [rho, N_Q, T, T_Q] = mm1(lambda, mu, k)
% simulacija servisnog sistema M/M/1 u GNU Octave

% oznake:
% k - broj obrađenih korisnika koji se posmatraju u simulaciji
% n - broj korisnika koji mogu ući u sistem
% i - brojač dolazaka
% j - brojač odlazaka
% IA - niz s vremenima međudolazaka
% A - niz s vremenima dolazaka
% W - niz s količinom posla
% D - niz s vremenima odlazaka
% t_Q - niz s vremenima čekanja
% t - niz s vremenima zadržavanja u sistemu
```

```
% t_stop - trenutak završetka simulacije
% rho - prosečna iskorišćenost servera
% N_Q - prosečan broj korisnika u čekaonici
% T - prosečno zadržavanje korisnika u sistemu
% T_Q - prosečno zadržavanje korisnika u čekaonici

% inicijalizacija i generisanje slučajnih promenljivih
n = 1.2*k;
i = 1;
j = 1;

IA = exprnd(1/lambda, n, 1);
W = exprnd(1/mu, n, 1);

A(1) = IA(1);
for m = 2:n
   A(m) = A(m-1)+IA(m);
endfor

D(1) = A(1)+W(1);
i = i+1;

% obrade događaja
while (j <= k)
   if (A(i) < D(j))
     % obrada dolaska
     i = i+1;
   else
     % obrada odlaska
     t_Q(j) = D(j)-A(j)-W(j);
     t(j) = D(j)-A(j);
     j = j+1;
       if (j != i)
         % sistem nije prazan
         D(j) = D(j-1)+W(j);
       else
         %sistem je prazan
         D(j) = A(j)+W(j);
       endif
   endif
endwhile

% obrada rezultata
t_stop = D(k)
rho = sum(W(1:k))/t_stop
N_Q = sum(t_Q)/t_stop
T = sum(t)/k
T_Q = sum(t_Q)/k

endfunction
```

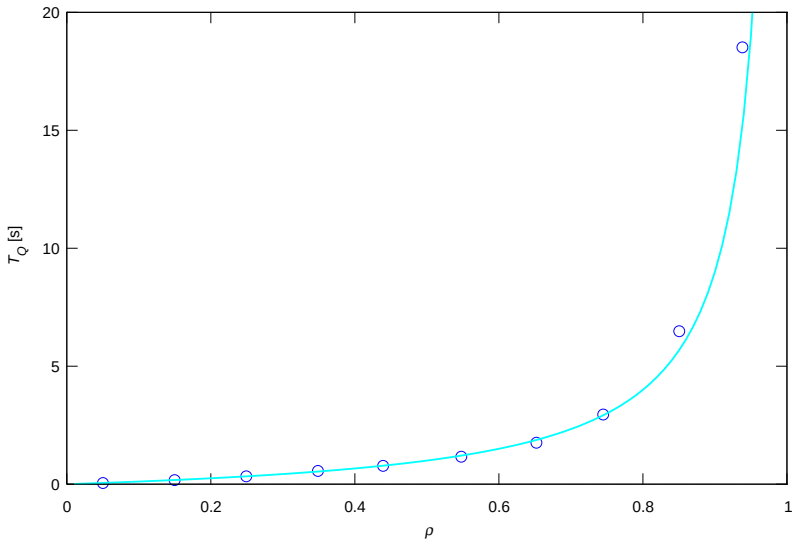
```
# mm1.py
# simulacija servisnog sistema M/M/1 u Pythonu

# poziv funkcije: python mm1.py lambd mu k
# "lambda" je rezervisana reč u Pythonu!

import random, sys, numpy as np

# oznake:
# k - broj obrađenih korisnika koji se posmatraju u simulaciji
# n - broj korisnika koji mogu ući u sistem
# i - brojač dolazaka
# j - brojač odlazaka
# IA - niz s vremenima međudolazaka
# A - niz s vremenima dolazaka
# W - niz s količinom posla
# D - niz s vremenima odlazaka
# t_Q - niz s vremenima čekanja
# t - niz s vremenima zadržavanja u sistemu
# t_stop - trenutak završetka simulacije
# rho - prosečna iskorišćenost servera
# N_Q - prosečan broj korisnika u čekaonici
# T - prosečno zadržavanje korisnika u sistemu
# T_Q - prosečno zadržavanje korisnika u čekaonici

# ulazni parametri
lambd = float(sys.argv[1])
mu = float(sys.argv[2])
k = int(sys.argv[3])

# inicijalizacija i generisanje slučajnih promenljivih
n = int(1.2*k)
i = 0
j = 0

IA = []
W = []
A = []
D = []
t_Q = []
t = []

for m in range(int(n)):
   IA.append(random.expovariate(lambd))
   W.append(random.expovariate(mu))
   A.append(sum(IA[0:m+1]))

D.append(A[0]+W[0])

i += 1
```

```
# obrada događaja
while (j < k):
   if (A[i] < D[j]):
      # obrada dolaska
      i += 1
   else:
      # obrada odlaska
      t_Q.append(D[j]-A[j]-W[j])
      t.append(D[j]-A[j])
      j += 1
      if (j != i):
         # sistem nije prazan
         D.append(D[j-1]+W[j])
      else:
         #sistem je prazan
         D.append(A[j]+W[j])


# obrada i ispis rezultata
t_stop = D[k-1]
rho = sum(W[0:k])/t_stop
N_Q = sum(t_Q)/t_stop
T = sum(t)/k
T_Q = sum(t_Q)/k
np.disp([rho, N_Q, T, T_Q])
```

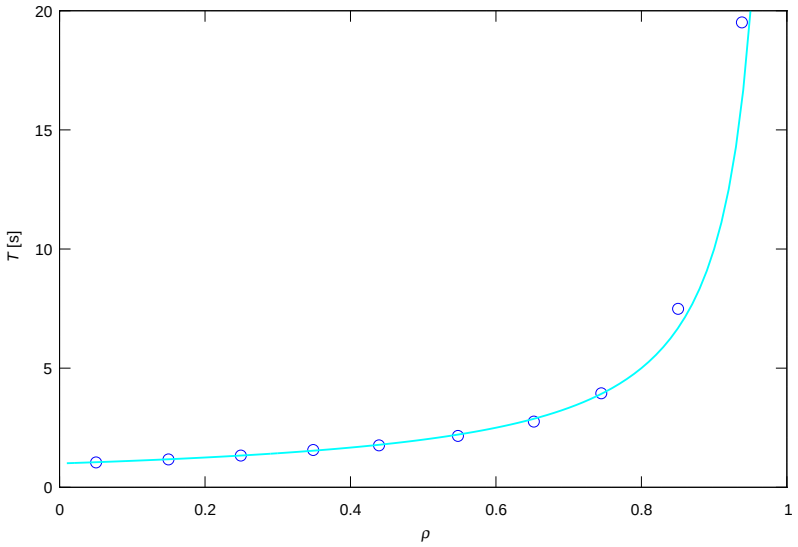
Simulacija je izvršavana za $\mu = 1\ \mathrm{s}^{-1}$, dok je vrednost λ menjana u opsegu od 0,05 do 0,95. Uslov za završetak je bio $k = 10\,000$. Rezultati su prikazani na narednim slikama, u vidu kružića. Radi poređenja, punim linijama su nacrtane teorijski dobijene vrednosti.



Slika 3.7: Prosečan broj korisnika u čekaonici.



Slika 3.8: Prosečno trajanje čekanja.



Slika 3.9: Prosečno zadržavanje korisnika u sistemu.

Vidimo da se za male vrednosti iskorišćenosti servera rezultati simulacije izuzetno dobro poklapaju s teorijskim, dok su za njene ekstremne vrednosti ($\rho \approx 1$) vidljiva odstupanja. Da bi se i u ovim uslovima ostvarila dobra tačnost, potrebno je produžiti trajanje simulacije, tj. broj obrađenih korisnika koji se posmatraju (k).

Poglavlje 4

Diskretizacija modela

U prethodnom poglavlju smo videli da je simulacija na digitalnom računaru metod izbora za analizu kontinualnih sistema u slučaju kada nije moguće – uopšte ili lako – doći do analitičkog rešenja. Pošto su kontinualni sistemi opisani diferencijalnim jednačinama, ovakav pristup implicitno podrazumeva diskretizaciju modela sistema i aproksimaciju diferencijalnih jednačina diferencnim. U nastavku ovoga poglavlja ćemo pokazati koje opasnosti vrebaju pri diskretnoj simulaciji kontinualnih sistema. Pored pitanja izbora koraka diferenciranja/integriranja, koje je povezano s teoremom o odabiranju, videćemo da se ovde javlja i problem stabilnosti numeričkog rešenja.

4.1 Numeričko diferenciranje

Neka je vrednost funkcije $f(x)$ poznata u ekvidistantnim tačkama[1] x_1, x_2, ... Najjednostavnije formule za diferenciranje se mogu dobiti iz definicije izvoda funkcije $f(x)$ u tački $x = x_i$,

$$f'(x_i) = \lim_{\Delta x \to 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}. \tag{4.1}$$

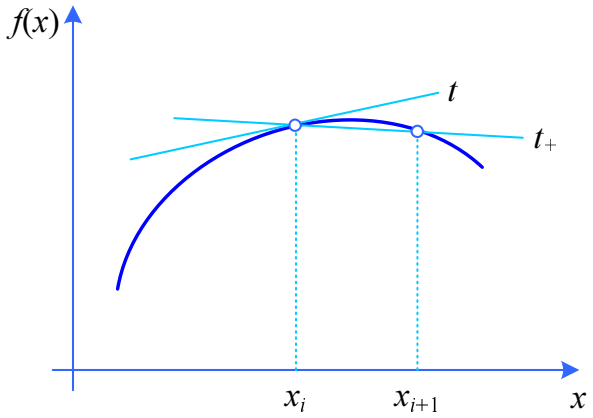
Usvojimo li da je $x_i + \Delta x = x_{i+1}$, odavde ćemo dobiti formulu za diferenciranje unapred, ili aproksimaciju izvoda prednjom razlikom:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{\Delta x}. \tag{4.2}$$

Geometrijska interpretacija diferenciranja unapred prikazana je na slici 4.1. Stvarnom izvodu funkcije $f(x)$ u tački x_i odgovara nagib linije t, dok nagib linije t_+ odgovara formuli za diferenciranje unapred.

Ako umesto Δx u (4.2) stavimo $-\Delta x$, dobićemo formulu za diferenciranje unazad, ili

[1] Kaže se i da je funkcija $f(x)$ tabelirana u datim tačkama.

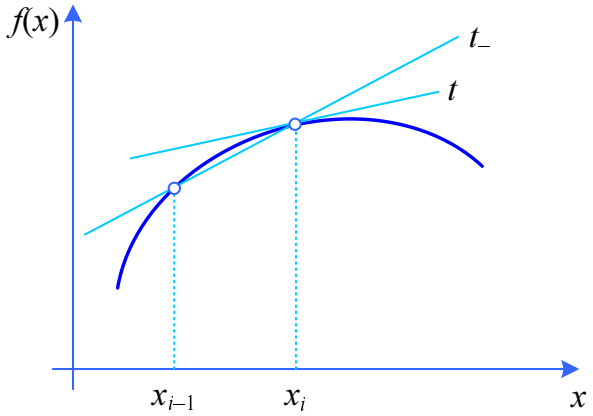


Slika 4.1: Diferenciranje unapred.

aproksimaciju izvoda zadnjom razlikom:

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{\Delta x}. \tag{4.3}$$

Geometrijska interpretacija diferenciranja unazad prikazana je na slici 4.2, gde ovakvom rezultatu odgovara nagib linije t_-.



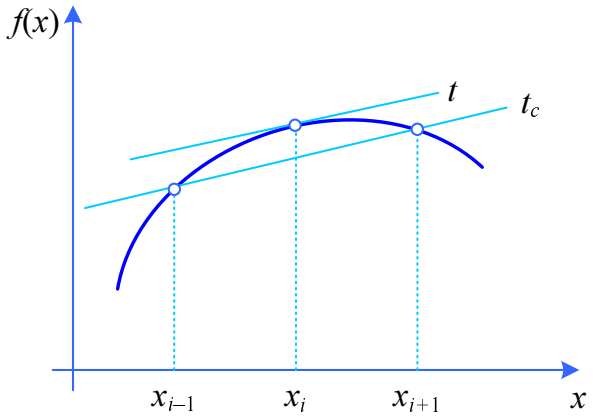
Slika 4.2: Diferenciranje unazad.

Sabiranjem formula za diferenciranje unapred i unazad i sređivanjem rezultata, dobijamo formulu za centralno diferenciranje, ili aproksimaciju izvoda centralnom razlikom:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x}. \tag{4.4}$$

Geometrijska interpretacija centralnog diferenciranja prikazana je na slici 4.3, gde ovakvom rezultatu odgovara nagib linije t_c. Sa slike vidimo, a to se može i formalno pokazati, da ovaj pristup daje rezultat koji je najbliži tačnom; nešto kasnije ćemo na numeričkom primeru videti i da je stabilnost rešenja dobijenog centralnim diferenciranjem (u smislu BIBO) najbolja.

Formule za izvode višeg reda možemo izvesti na sličan način. Najjednostavnije apro-



Slika 4.3: Centralno diferenciranje.

ksimacije drugog izvoda tako su, redom:

$$f''(x_i) \approx \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{(\Delta x)^2}, \tag{4.5}$$

$$f''(x_i) \approx \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{(\Delta x)^2}, \tag{4.6}$$

$$f''(x_i) \approx \frac{f(x_{i+2}) - 2f(x_i) + f(x_{i-1})}{(\Delta x)^2}. \tag{4.7}$$

4.2 Numeričko integriranje

Razmotrićemo najjednostavnije koračne formule za numeričko integriranje, koje se direktno izvode iz formula za numeričko diferenciranje[2].

Neka je ponovo funkcija $f(x)$ tabelirana u ekvidistantnim tačkama $x_1, x_2, \ldots$ Potrebno je izračunati integral

$$I = \int_{x_1}^{x_{i+1}} f(x)\,\mathrm{d}x. \tag{4.8}$$

Ovaj integral možemo napisati u sledećem obliku:

$$I = I_{i+1} = \int_{x_1}^{x_i} f(x)\,\mathrm{d}x + \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x =$$

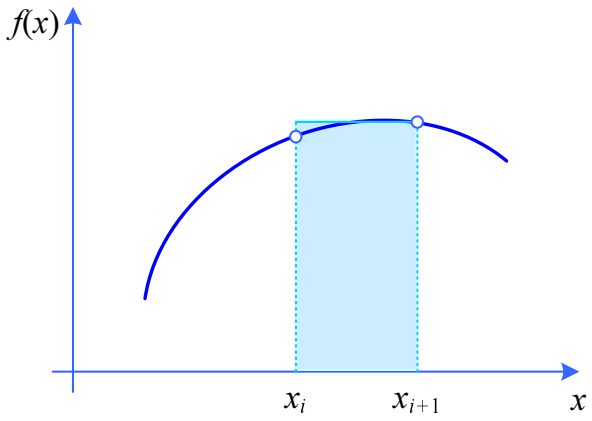
$$= I_i + \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x. \tag{4.9}$$

[2]Daleko bolji rezultati u praksi se postižu primenom tzv. kvadraturnih formula, npr. Gauss--Legendreovih.

Formulu za integriranje unapred dobijamo ako usvojimo da je $f(x) = f(x_{i+1})$:

$$I_{i+1} = I_i + f(x_{i+1})\Delta x. \tag{4.10}$$

Geometrijska interpretacija integriranja unapred data je na slici 4.4.



Slika 4.4: Integriranje unapred.

Nije teško pokazati da je integriranje unapred ekvivalentno diferenciranju unazad: iz (4.10) je

$$\frac{I_{i+1} - I_i}{\Delta x} = f(x_{i+1}), \tag{4.11}$$

pa je i

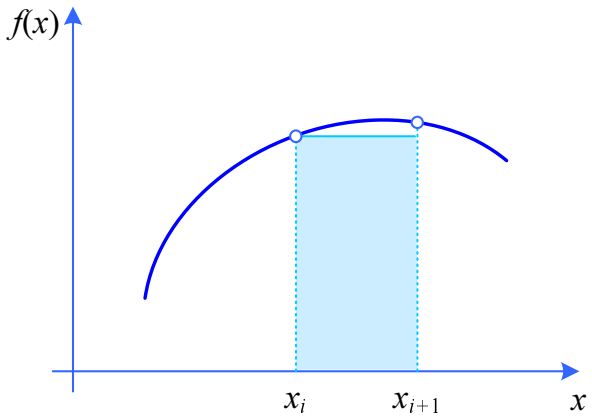
$$\frac{I_i - I_{i-1}}{\Delta x} = f(x_i), \tag{4.12}$$

što odgovara izrazu (4.3).

Formulu za integriranje unazad dobijamo ako u (4.9) usvojimo da je $f(x) = f(x_i)$:

$$I_{i+1} = I_i + f(x_i)\Delta x. \tag{4.13}$$

Geometrijska interpretacija integriranja unazad data je na slici 4.5.



Slika 4.5: Integriranje unazad.

Integriranje unazad je ekvivalentno diferenciranju unapred. Iz (4.13) je

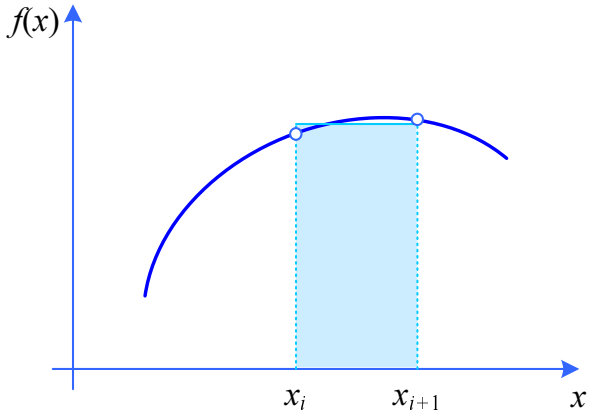
$$\frac{I_{i+1} - I_i}{\Delta x} = f(x_i), \tag{4.14}$$

što odgovara izrazu (4.2).

Formulu za centralno integriranje, poznatu i kao trapezno pravilo, dobićemo sabiranjem formula za integriranje unapred i unazad i sređivanjem rezultata:

$$I_{i+1} = I_i + \frac{f(x_i) + f(x_{i+1})}{2}\Delta x. \tag{4.15}$$

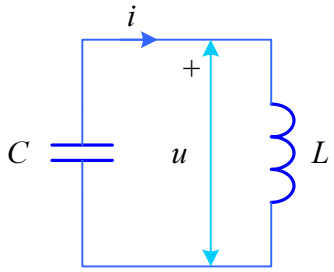
Geometrijska interpretacija centralnog integriranja data je na slici 4.6. Ponovo intuitivno vidimo da ovaj pristup daje najtačniji rezultat.



Slika 4.6: Centralno integriranje.

4.3 Primer

„Kvalitet" navedenih formula ćemo ispitati na primeru koji je samo na prvi pogled trivijalan. Posmatraćemo paralelno LC kolo, prikazano na slici 4.7.



Slika 4.7: Paralelno LC kolo.

Poznat je početni uslov $i(t = 0) = i_0$, $u(t = 0) = 0$. *Znamo* da je kolo opisano integralnim

$$u(t) = -\frac{1}{C}\int_0^t i(\tau)\,\mathrm{d}\tau, \tag{4.16}$$

$$i(t) = \frac{1}{L} \int_0^t u(\tau)\, \mathrm{d}\tau, \tag{4.17}$$

ili diferencijalnim jednačinama,

$$u(t) = L\, \frac{\mathrm{d}i(t)}{\mathrm{d}t}, \tag{4.18}$$

$$i(t) = -C\, \frac{\mathrm{d}u(t)}{\mathrm{d}t}. \tag{4.19}$$

Takođe *znamo* kako izgleda rešenje ovih jednačina: pošto u kolu nema niti izvora, niti potrošača (komponente su bez gubitaka), očekujemo prostoperiodični režim. Ovo je redak slučaj sistema čije nam je analitičko rešenje poznato, pa će nam ono služiti kao reper za poređenje numeričkih metoda.

Navedene jednačine ćemo rešiti numeričkim metodama koje smo razmatrali u prethodnim odeljcima. Iako bismo to mogli, nećemo kombinovati po dve jednačine da bismo dobili jednu drugog reda, već ćemo rešavati sistem integralnih/diferencijalnih jednačina prvog reda.

Kao i u prethodnim formulama, korak integracije ćemo označiti sa Δt. Izračunavanja ćemo vršiti u programu GNU Octave. Usvojićemo sledeće vrednosti parametara: $L = 1/(2\pi)$ H, $C = 1/(2\pi)$ F, $\Delta t = T/50$, gde je $T = 2\pi\sqrt{LC} = 1$ s period oscilacija. Pošto indeksiranje nizova kreće od cifre 1, u formulama u nastavku su usvojene oznake $u(n) = u((n-1)\Delta t)$ i $i(n) = i((n-1)\Delta t)$. Iz istog razloga, početni uslovi su `i(1) = 1` ($i_0 = 1$ A) i `u(1) = 0` ($u_0 = 0$ V).

4.3.1 Integriranje unapred

Primenom formule za integriranje unapred, (4.10), na (4.16) i (4.17), dobijamo

$$u(n) = u(n-1) - \frac{1}{C}\, i(n)\Delta t, \tag{4.20}$$

$$i(n) = i(n-1) + \frac{1}{L}\, u(n)\Delta t. \tag{4.21}$$

Uvrštavanjem (4.21) u (4.20), nakon sređivanja dobijamo

$$u(n) = \frac{u(n-1) - \dfrac{1}{C}\, i(n-1)\Delta t}{1 + \dfrac{(\Delta t)^2}{LC}}. \tag{4.22}$$

Formule (4.22) i (4.21) predstavljaju rešenje posmatranog kola metodom integriranja unapred.

4.3.2 Integriranje unazad

Neposrednom primenom formule za integriranje unazad (4.13) na (4.16) i (4.17), dobijamo

$$u(n) = u(n-1) - \frac{1}{C}\, i(n-1)\Delta t, \tag{4.23}$$

$$i(n) = i(n-1) + \frac{1}{L}\, u(n-1)\Delta t. \tag{4.24}$$

4.3.3 Centralno integriranje

Primenom formule za centralno integriranje, (4.15), na (4.16) i (4.17), dobijamo

$$u(n) = u(n-1) - \frac{1}{C}\, \frac{i(n-1)+i(n)}{2}\Delta t, \tag{4.25}$$

$$i(n) = i(n-1) + \frac{1}{L}\, \frac{u(n-1)+u(n)}{2}\Delta t. \tag{4.26}$$

Uvrstimo li (4.26) u (4.25) i sredimo li izraz, dobićemo

$$u(n) = \frac{\left[1 - \dfrac{(\Delta t)^2}{4LC}\right] u(n-1) - \dfrac{\Delta t}{C}\, i(n-1)}{1 + \dfrac{(\Delta t)^2}{4LC}}. \tag{4.27}$$

Formule (4.27) i (4.26) predstavljaju rešenje LC kola metodom centralnog integriranja.

4.3.4 Diferenciranje unapred

Primenimo li formulu za diferenciranje unapred (4.2) na jednačine (4.18) i (4.19), dobićemo

$$u(n) = L\, \frac{i(n+1)-i(n)}{\Delta t}, \tag{4.28}$$

$$i(n) = -C\, \frac{u(n+1)-u(n)}{\Delta t}. \tag{4.29}$$

Umanjimo li indekse za 1, imaćemo

$$u(n-1) = L\, \frac{i(n)-i(n-1)}{\Delta t}, \tag{4.30}$$

$$i(n-1) = -C\, \frac{u(n)-u(n-1)}{\Delta t}. \tag{4.31}$$

Odavde je rešenje LC kola metodom diferenciranja unapred:

$$u(n) = u(n-1) - \frac{\Delta t}{C}\, i(n-1), \tag{4.32}$$

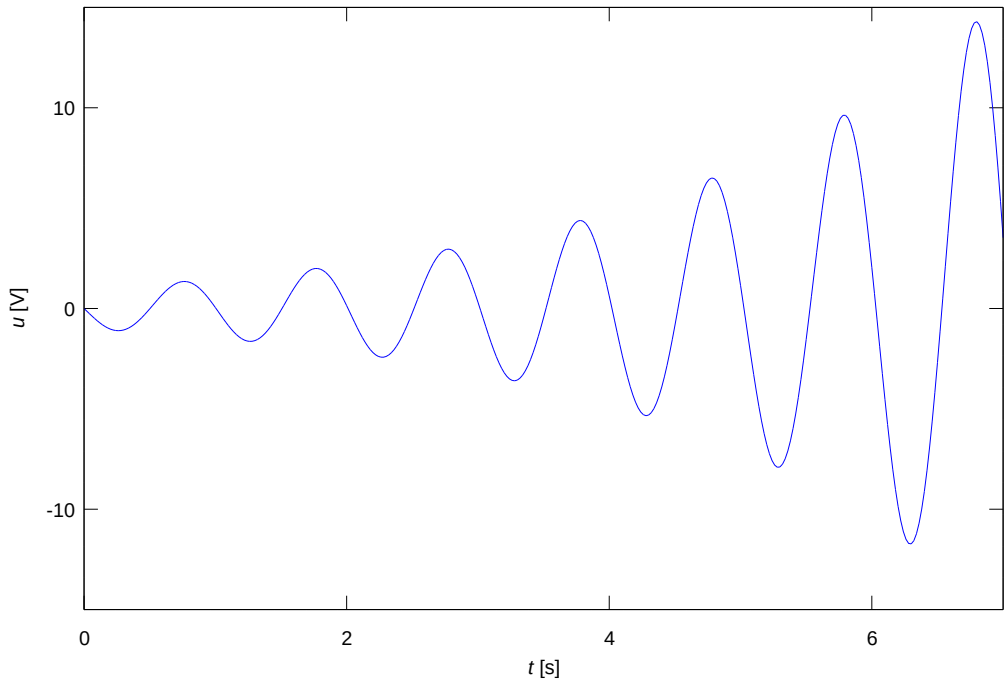
$$i(n) = i(n-1) + \frac{\Delta t}{L}\, u(n-1), \tag{4.33}$$

što odgovara formulama koje su dobijene integracijom unazad, (4.23) i (4.24).

Octave programski kod koji rešava kolo na ovaj način za prvih 7 perioda je

```
L = 1/(2*pi);
C = 1/(2*pi);
T = 2*pi*sqrt(L*C);
dt = T/50;
i(1) = 1;
u(1) = 0;
for n = 2:(7/dt+1)
    u(n) = u(n-1) - (dt/C)*i(n-1);
    i(n) = i(n-1) + (dt/L)*u(n-1);
end
t = 0:dt:7;
plot(t, u), xlabel("{\itt} [s]"), ylabel("{\itu} [V]"), axis([0 7 -15 15])
```

Grafik ovako proračunatog napona je prikazan na slici 4.8. Već na prvi pogled uočavamo da njegova amplituda nije tačna i da je rešenje nestabilno, a pažljivijim posmatranjem grafika možemo zaključiti i da je period ovih oscilacija veći od stvarnog.



Slika 4.8: Napon u kolu, dobijen integriranjem unazad/diferenciranjem unapred.

4.3.5 Diferenciranje unazad

Primenom formule za diferenciranje unapred na diferencijalne jednačine kola, dobijamo

$$u(n) = L\,\frac{i(n) - i(n-1)}{\Delta t}, \tag{4.34}$$

$$i(n) = -C\,\frac{u(n) - u(n-1)}{\Delta t}. \tag{4.35}$$

Uvrštavanjem (4.35) u (4.34), nakon sređivanja dobijamo

$$u(n) = \frac{u(n-1) - \dfrac{1}{C}\,i(n-1)\Delta t}{1 + \dfrac{(\Delta t)^2}{LC}}. \tag{4.36}$$

Izrazi (4.36) i (4.35) redom odgovaraju formulana dobijenim integracijom unapred, (4.22) i (4.21).

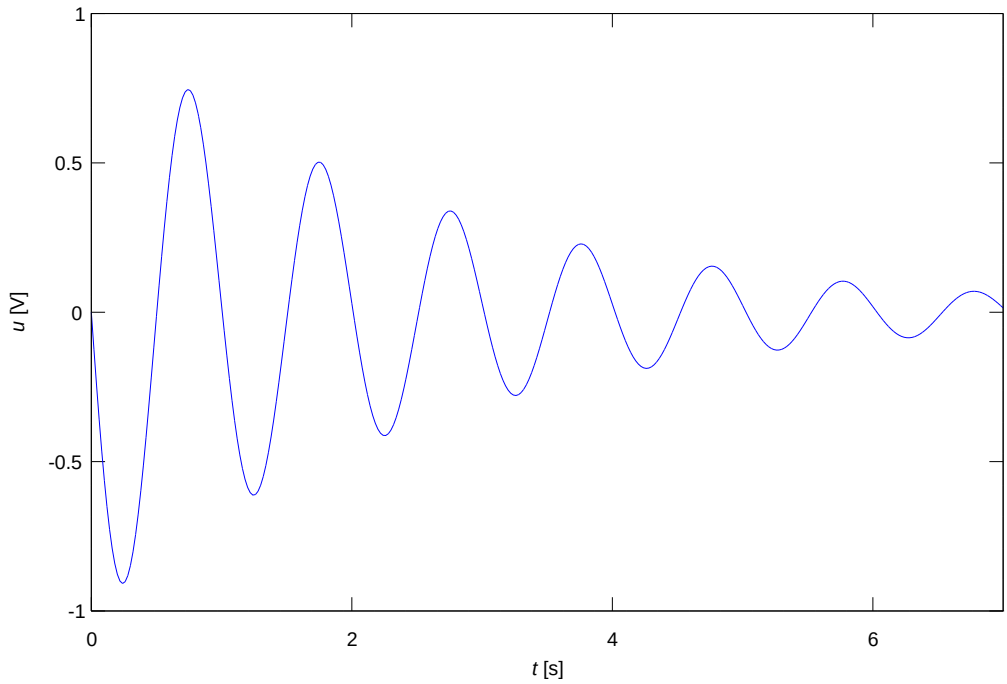
Octave programski kod koji rešava kolo na ovaj način je

```
L = 1/(2*pi);
C = 1/(2*pi);
T = 2*pi*sqrt(L*C);
dt = T/50;
i(1) = 1;
u(1) = 0;
for n = 2:(7/dt + 1)
   u(n) = (u(n-1) - (dt/C)*i(n-1))/(1+(dt)^2/(L*C));
   i(n) = i(n-1) + (dt/L)*u(n);
end
t = 0:dt:7;
plot(t, u), xlabel("{\itt} [s]"), ylabel("{\itu} [V]"), axis([0 7 -1 1])
```

Grafik napona je prikazan na slici 4.9. Ni ovo rešenje nije dobro, jer smo sada dobili opadajuće oscilacije; ono što se teže uočava je da one i ovaj put imaju nešto veći period od tačnog.

4.3.6 Centralno diferenciranje

Sistem diferencijalnih jednačina (4.18) i (4.19) rešićemo centralnim diferenciranjem tako što ćemo sabrati rešenja koja smo dobili za diferenciranje unapred i unazad i



Slika 4.9: Napon u kolu, dobijen integriranjem unapred/diferenciranjem unazad.

potom srediti izraze. Na taj način, dobijamo

$$u(n) = \frac{\left[1 - \frac{(\Delta t)^2}{4LC}\right] u(n-1) - \frac{\Delta t}{C}\, i(n-1)}{1 + \frac{(\Delta t)^2}{4LC}}, \tag{4.37}$$

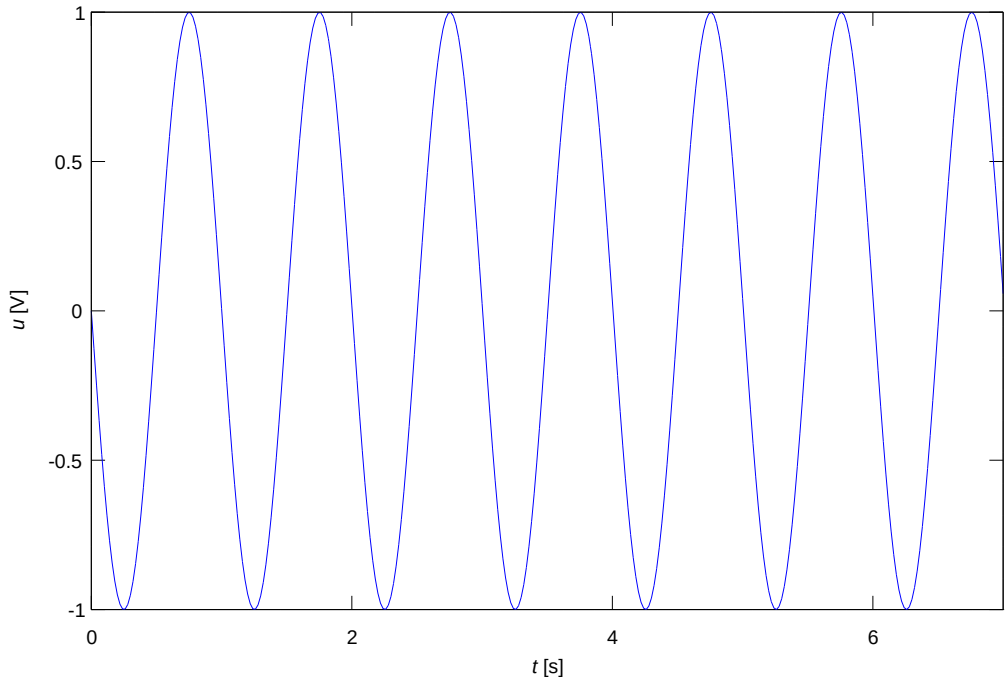
$$i(n) = i(n-1) + \frac{1}{L}\,\frac{u(n-1) + u(n)}{2}\Delta t. \tag{4.38}$$

Ovi izrazi su identični izrazima (4.27) i (4.26), koje smo u odeljku 4.3.3 dobili metodom centralnog integriranja.

Octave programski kod koji rešava kolo na ovaj način je

```
L = 1/(2*pi);
C = 1/(2*pi);
T = 2*pi*sqrt(L*C);
dt = T/50;
i(1) = 1;
u(1) = 0;
for n = 2:(7/dt+1)
   u(n) = ((1-(dt^2)/(4*L*C))*u(n-1)-dt/C*i(n-1))/(1+(dt^2)/(4*L*C));
   i(n) = i(n-1)+dt/(2*L)*u(n-1)+dt/(2*L)*u(n);
end
t = 0:dt:7;
plot(t, u), xlabel("{\itt} [s]"), ylabel("{\itu} [V]"), axis([0 7 -1 1])
```

Grafik napona je dat na slici 4.10. Konačno smo dobili rešenje koje odgovara analitičkom, a to je prostoperiodični režim amplitude 1 V i perioda (veoma približno) 1 s.



Slika 4.10: Napon u kolu, dobijen centralnim integriranjem/diferenciranjem.

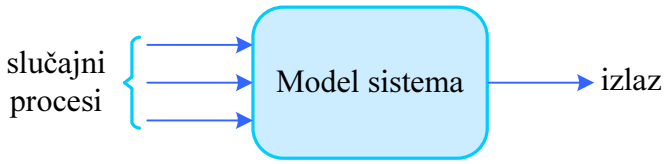
Poglavlje 5

Monte Carlo simulacija

U ovome poglavlju ćemo se upoznati s tehnikom simulacije koja se najčešće primenjuje u telekomunikacijama. Ona koristi slučajne brojeve, pa je nazvana Monte Carlo, što je jasna asocijacija na igre na sreću. U današnjem obliku prvi put je primenjena tokom II svetskog rata, prilikom razvoja atomske bombe.

5.1 Princip Monte Carlo simulacije

Monte Carlo (MC) simulacija se zasniva na ponavljanju simuliranih slučajnih eksperimenata. Najčešće se implementira tako što se analizirani sistem pobuđuje povorkama slučajnih brojeva koje emuliraju fizičke ulazne procese.

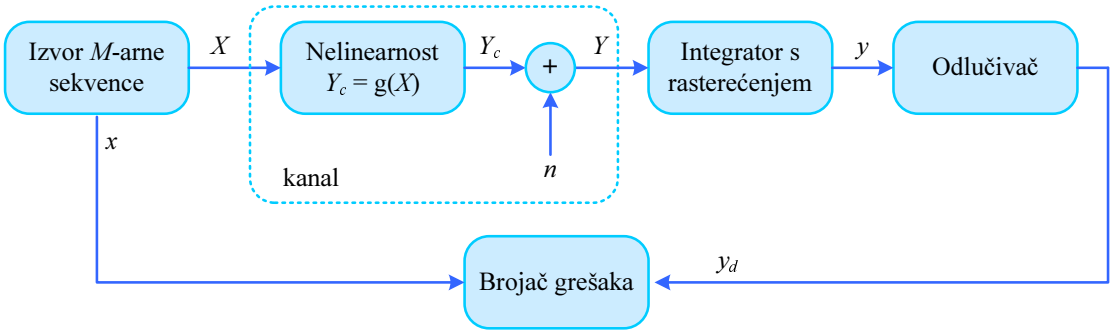


Slika 5.1: Princip Monte Carlo simulacije.

Ako su ulazi sistema slučajni procesi, tada će to biti i njegov izlaz. Rezultat simulacije se *ocenjuje* iz izlaznog slučajnog procesa nekom od statističkih metoda.

MC simulacija se može primeniti za analizu kako determinističkih, tako i stohastičkih modela sistema.

Kao primer primene MC simulacije u telekomunikacijama, posmatrajno sistem za prenos M-arnih digitalnih signala u osnovnom opsegu učestanosti. Potrebno je odrediti verovatnoću greške po simbolu, u uslovima nelinearnog kanala u kome deluje aditivni šum. Blok-šema simulacionog modela prikazana je na slici 5.2.



Slika 5.2: Simulacioni model sistema za prenos signala u osnovnom opsegu učestanosti.

U nastavku ćemo razmotriti korake koje bi obuhvatala Monte Carlo simulacija ovakvog sistema i uporedo ćemo pokazati kako bi se oni realizovali u GNU Octave i Pythonu.

Na početku se generiše N simbola npr. kvaternarnog ulaznog signala i mN odbiraka Gaussovog šuma. Primetimo da se po svakom simbolu generiše m odbiraka šuma, pa ćemo i za svaki simbol uzeti po m odbiraka.

GNU Octave:

```
x = randint(1, N, [0, 3])*2 - 3;
X = [];
for i = 1:N
   X = [X, x(i)*ones(1, m)];
endfor

n = sigma*randn(1, N*m);
```

Python:

```
from pylab import *

x = randint(0, 4, N)*2 - 3
X = repeat (x, m)
n = sigma*randn(N*m)
```

Signal se potom propusti kroz kanal čiji je model poznat:

```
Yc = g(X);
Y = Yc + n;
```

```
Yc = g(X)
Y = Yc + n
```

Integriranje s rasterećenjem modeliramo računanjem srednje vrednosti primljenog signala na intervalu trajanja jednog simbola:

```
y = [];
for i = 1:N
   y = [y, mean(Y((i-1)*m+1:i*m))];
endfor
```

```
y = []
for i in range(N)
   y.append(mean(Y[i*m:(i+1)*m-1]))
```

U odlučivaču se signal poredi s pragovima, na osnovu čega se donosi odluka:

```
for i = 1:N
   if (y(i) <= -2)
      yd(i) = -3;
```

```
yd = []
for yy in y:
   if yy <= -2:
      yd.append(-3)
```

```
   elseif ((-2 < y(i)) & (y(i) <= 0))
      yd(i) = -1;
   elseif ((0 < y(i)) & (y(i) <= 2))
      yd(i) = 1;
   else
      yd(i) = 3;
   endif
endfor
```

```
elif -2 < yy <= 0:
   yd.append(-1)
elif 0 < yy <= 2:
   yd.append(1)
else:
   yd.append(3)
```

Konačno, potrebno je prebrojati pogrešno primljene simbole i oceniti verovatnoću greške:

```
SER = sum(yd!=x)/N;
```

```
SER = sum(yd!=x)/N
```

S metodama obrade rezultata simulcije i ocene parametara performansi analiziranog sistema upoznaćemo se u narednom poglavlju. Da bismo upotpunili započeti primer, sada ćemo reći da se često primenjuje tzv. metod nezavisnih replikacija, po kome se izvrši više serija nezavisnih ponavljanja simulacije. Nezavisnost se ostvaruje npr. različitim početnim uslovima, ili, u našem primeru, različitim sekvencama korisnog signala i šuma. Za svaku seriju replikacija se oceni traženi izlazni parametar (npr. verovatnoća greške), a konačan rezultat se dobije usrednjavanjem ovih „podrezultata" po svim serijama.

Intuitivno je jasno da će tačnost rezultata dobijenog Monte Carlo simulacijom zavisiti od broja replikacija statističkog eksperimenta, odnosno, u gornjem primeru, od broja generisanih simbola. Videćemo u odeljku 6.2.8 na strani 73 da rezultat Monte Carlo simulacije sporo konvergira, što znači da je u opštem slučaju potrebno mnogo replikacija, pa su Monte Carlo simulacije dugotrajne. U poglavlju 7 ćemo videti kako se može proceniti potreban broj replikacija da bi se postigla željena tačnost ocene izlaznog parametra.

5.2 Kvazianalitička Monte Carlo simulacija

U nekim primenama nije neophodno da se simuliraju svi ulazni procesi, niti da se simuliraju svi slučajni procesi u sistemu, već se njihovi efekti mogu uzeti u obzir korišćenjem analitičkih tehnika. Ovo je osnova kvazianalitičke (QA) ili parcijalne Monte Carlo simulacije.

Da bismo ilustrovali ovaj princip, zadržimo se na primeru sistema za prenos digitalnog signala u osnovnom opsegu učestanosti. Ako je prijemnik idealan i ako je šum na njegovom ulazu aditivan i Gaussov, tada nema potrebe da simuliramo efekat šuma, jer će šum na ulazu sklopa za odlučivanje takođe biti aditivan i Gaussov. Dovoljno je simulirati samo prenos signala kroz nelinearni kanal, dok se efekat šuma može posmatrati analitički, preko poznatog izraza za verovatnoću greške.

Primenom kvazianalitičkog pristupa može se smanjiti količina podataka koje treba obraditi da bi se postigla jednaka tačnost kao za osnovnu Monte Carlo simulaciju.

5.3 (Neobičan?) primer

Pokazaćemo kako se Monte Carlo metoda može primeniti za izračunavanje određenih integrala.

Neka je

$$I = \int_a^b g(x)\,\mathrm{d}x \tag{5.1}$$

integral koji treba izračunati. Formalno, možemo smatrati da (5.1) predstavlja ulazno-izlaznu relaciju sistema pobuđenog signalom x.

Setimo li se teorema integralnog računa, u intervalu $[a,b]$ postoji tačka X_0 za koju je $(b-a)g(X_0) = I$. Problem je u tome što ne znamo gde se tačno ta tačka nalazi; stoga ćemo primeniti Monte Carlo simulaciju.

Neka je X slučanja promenljiva, uniformno raspodeljena na $[a,b]$. Definišimo novu slučajnu promenljivu Y kao

$$Y = (b-a)g(X). \tag{5.2}$$

Odredimo očekivanje slučajne promenljive Y. Iz (5.2) je

$$\begin{aligned} \mathrm{E}Y &= \mathrm{E}(b-a)g(X) = \\ &= (b-a)\mathrm{E}g(X) = \\ &= (b-a)\int_{-\infty}^{+\infty} g(x)f_X(x)\,\mathrm{d}x, \end{aligned} \tag{5.3}$$

gde je $f_X(x)$ funkcija gustine verovatnoće slučajne promenljive X,

$$f_X(x) = \begin{cases} \dfrac{1}{b-a}, & x \in [a,b] \\ 0, & \text{inače} \end{cases}. \tag{5.4}$$

Uvrštavanjem u (5.3), dobijamo

$$\begin{aligned} \mathrm{E}Y &= (b-a)\int_a^b \frac{g(x)}{b-a}\,\mathrm{d}x = \\ &= \int_a^b g(x)\,\mathrm{d}x = I. \end{aligned} \tag{5.5}$$

Ovaj rezultat je osnov za izračunavanje integrala metodom Monte Carlo.

Kao primer, posmatrajmo integral

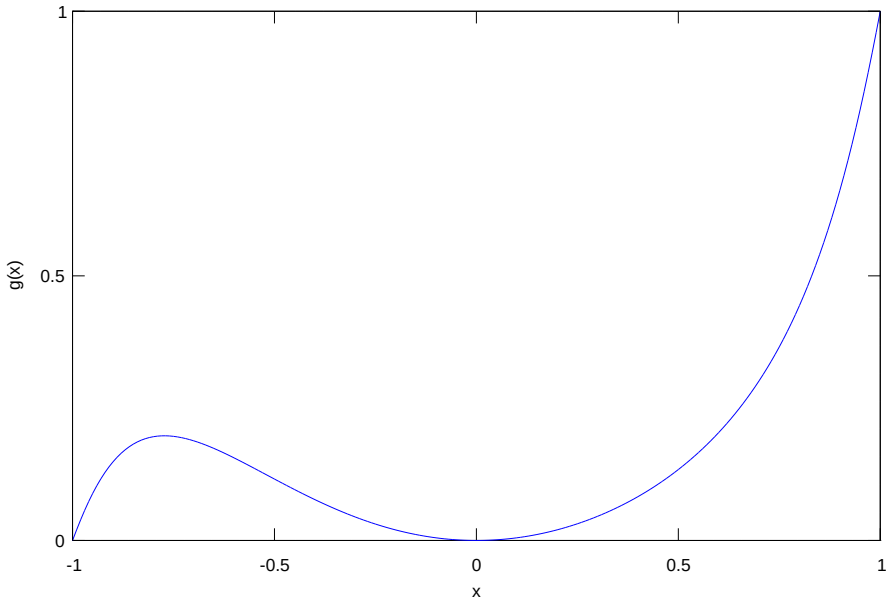
$$J = \int_{-1}^{1} \frac{x^2}{1 + (1 - x^2)^x} \, \mathrm{d}x. \tag{5.6}$$

Da bismo stekli predstavu o podintegralnoj funkciji, nacrtajmo njen grafik. U programu GNU Octave, kucajmo

```
> function y = g(x)
> y = x.^2./(1+(1-x.^2).^x);
> endfunction
```

Ovim smo definisali podintegralnu funkciju y. Njen grafik na intervalu integracije nacrtaćemo komandama

```
> x = -1:0.01:1;
> w = g(x);
> plot(x, w)
```



Slika 5.3: Grafik podintegralne funkcije.

Grafik smo mogli nacrtati i jednostavnije, bez definisanja funkcije `g(x)`, tako što bismo posle definisanja vektora `x` kucali `y = x.^2./(1+(1-x.^2).^x)` i potom `plot(x, y)`. Ono što sada izgleda kao duži pristup, uštedeće nam vreme za izračunavanja koja će uskoro uslediti.

Oznake na ose postavljamo komandama

```
> xlabel("x")
> ylabel("g(x)")
```

a grafik ćemo sačuvati kao npr. eps datoteku naredbom

```
> print -deps d:\grafik1.eps
```

Iako se na prvi pogled to ne bi reklo, J je analitički rešiv. Može se pokazati da važi sledeći opšti rezultat: ako su $f_1(x)$ i $f_2(x)$ proizvoljne parne funkcije i $g(x)$ proizvoljna neparna funkcija, tada je

$$\int_{-a}^{a} \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}} \,\mathrm{d}x = \int_{0}^{a} f_1(x) \,\mathrm{d}x. \tag{5.7}$$

Dokaz je dat u odeljku 5.4.1. U našem slučaju će biti

$$J = \int_{0}^{1} x^2 \,\mathrm{d}x = \frac{1}{3}. \tag{5.8}$$

Numeričkom integracijom po Gauss-Legendreovoj kvadraturnoj formuli, potvrđujemo ovaj rezultat:

```
> quad("g", -1, 1)
ans = 0.33333
```

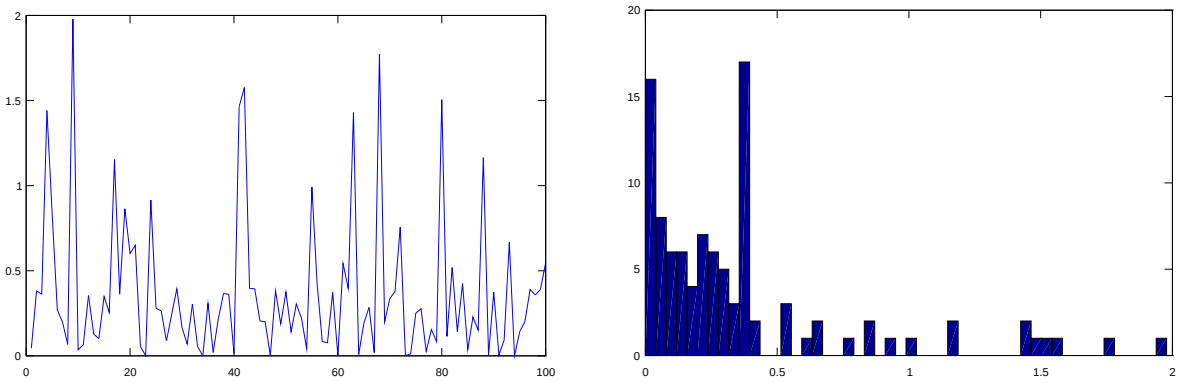
Izvršimo prvu seriju Monte Carlo simulacija, sa $N = 100$ ponavljanja:

```
N = 100;
a = -1;
b = 1;
X = unifrnd(a, b, N, 1);
Y = (b-a)*g(X);
mean(Y), var(Y)
```

U jednoj realizaciji ovakvog eksperimenta, dobili smo srednju vrednost uzorka 0,29787 i njegovu varijansu 0,13046, što nije naročito dobro. Primenićemo, stoga, metod nezavisnih replikacija i ponovićemo seriju N eksperimenata 100 puta. Na slici 5.4, prikazani su grafik i histogram srednje vrednosti ovako dobijenih rezultata. Histogram je nacrtan naredbom `hist(Y, n)`, čiji su argumenti redom vektor uzorka i broj „stubića".

Srednja vrednost nakon 100 ponavljanja, odnosno ukupno $100 \cdot 100$ eksperimenata je 0,32933, što je već blisko tačnoj vrednosti. Opisanu proceduru smo ponovili za $N = 1\,000$ i $N = 10\,000$ i dobijene rezultate prikazali u tabeli 5.1.

Iz tabele vidimo da rezultati Monte Carlo simulacija sporo i, u opštem slučaju *nemonotono* konvergiraju ka tačnoj vrednosti; to je jedan od razloga zbog čega Monte Carlo neće biti metoda izbora za numeričko rešavanje integrala.



Slika 5.4: Grafik (levo) i histogram (desno) srednje vrednosti serije Monte Carlo simulacija.

Tabela 5.1: Rezultati simulacija.

N	prva serija	usrednjeno na 100 serija
100	0,29787	0,32933
1 000	0,32797	0,33350
10 000	0,33079	0,33364

5.4 Dodaci

5.4.1 Dokaz izraza (5.7)

Posmatrajmo integral

$$I = \int_{-a}^{a} \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}} \, \mathrm{d}x, \tag{5.9}$$

u kome su $f_1(x)$ i $f_2(x)$ proizvoljne parne funkcije i $g(x)$ proizvoljna neparna funkcija. Razdvojimo li interval integriranja na dva simetrična podintervala, imaćemo

$$I = \int_{-a}^{0} \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}} \, \mathrm{d}x + \int_{0}^{a} \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}} \, \mathrm{d}x. \tag{5.10}$$

Uvedimo u prvom integralu smenu $t = -x$:

$$I = \int_{a}^{0} \frac{f_1(-t)}{1 \pm f_2(-t)^{g(-t)}} \, \mathrm{d}(-t) + \int_{0}^{a} \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}} \, \mathrm{d}x. \tag{5.11}$$

Imajući u vidu da su $f_1(x)$ i $f_2(x)$ parne funkcije, a da je $g(x)$ neparna, dalje će biti

$$I = \int_0^a \frac{f_1(t)}{1 \pm f_2(t)^{-g(t)}}\,\mathrm{d}t + \int_0^a \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}}\,\mathrm{d}x =$$
$$= \int_0^a \frac{f_1(t)\left[\pm f_2(t)^{g(t)}\right]}{\pm f_2(t)^{g(t)} + 1}\,\mathrm{d}t + \int_0^a \frac{f_1(x)}{1 \pm f_2(x)^{g(x)}}\,\mathrm{d}x. \tag{5.12}$$

Odavde se konačno dobija

$$I = \int_0^a f_1(x)\,\mathrm{d}x, \tag{5.13}$$

što je i trebalo dokazati.

5.4.2 (Neobičan?) primer u Pythonu

U nastavku je dat programski kod u Pythonu kojim se rade primeri iz odeljka 5.3.

```
from pylab import *
from scipy import integrate

# definisanje podintegralne funkcije
def g(x):
   y = x**2/(1+(1-x**2)**x)
   return y

# crtanje grafika
x = arange(-1, 1, 0.01) # ekvivalentno sa x = array(range(-100, 100))*0.01
w = g(x)
plot(x, w)
show()
xlabel("x")
ylabel("g(x)")

# snimanje grafika
savefig("d:\grafik1.eps")

# numeričko integriranje
integrate.quad(g, -1, 1)
# rezultati su vrednost integrala i procenjena maksimalna apsolutna greška

# Monte Carlo integriranje
N = 100
a = -1
b = 1
```

```
X = uniform(a, b, N)
Y = (b-a)*g(X)
mean(Y), var(Y)

# histogram s 20 stubića
figure() # otvara novi prozor
hist(Y, 20)
```

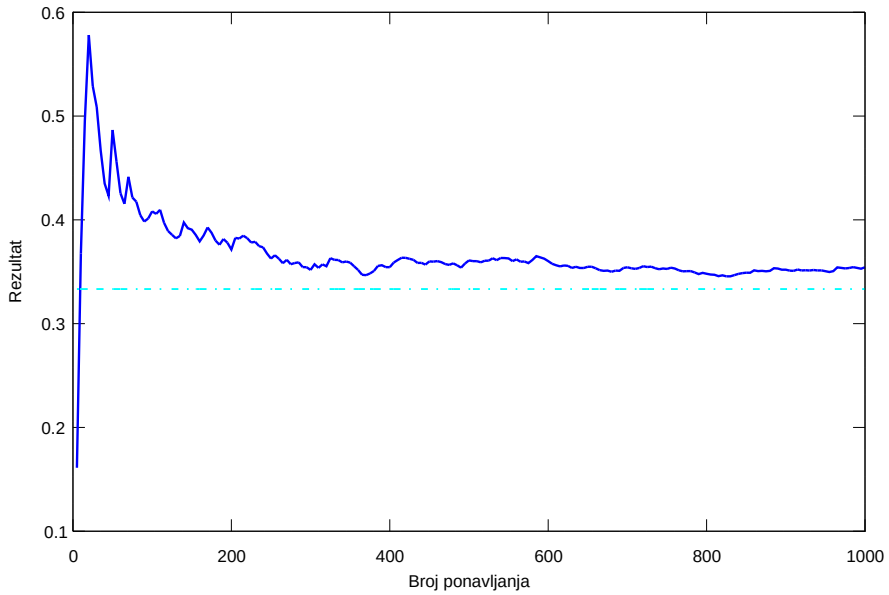
Poglavlje 6

Obrada rezultata simulacije

U ovome poglavlju ćemo se upoznati sa statističkom obradom rezultata serije simulacija: videćemo kako se identifikuje stacionarni režim, kako se ocenjuju parametri raspodele i određuju intervali poverenja, a izvršićemo i statističke testove.

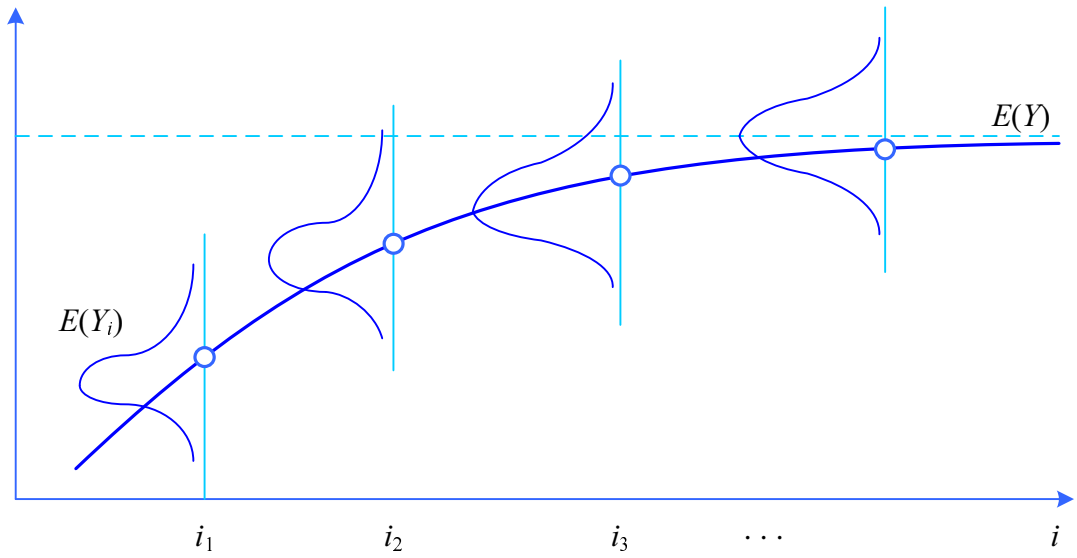
6.1 Prelazni i ustaljeni režim

Izlaz simulacije, Y, predstavlja slučajnu promenljivu čiji statistički parametri zavise od broja ponavljanja simulacije i od pretpostavljenih početnih uslova. Na narednoj slici je prikazana zavisnost rezultata izračunavanja integrala (5.6) od broja tačaka u Monte Carlo metodi.



Slika 6.1: Promena rezultata simulacije. Tačna vrednost je predstavljena isprekidanom linijom.

U opštem slučaju, s povećanjem broja replikacija, i, menjaju se i funkcija gustine verovatnoće i matematičko očekivanje izlaza simulacije. Ova zavisnost je kvalitativno prikazana na slici 6.2.



Slika 6.2: Promena statistike izlaza.

Neka je I početni uslov za koji je izvršeno i replika simulacije. Intuitivno je jasno da izlaz simulacije zavisi i od početnog uslova, pa možemo definisati uslovnu funkciju raspodele

$$F_i(y|I) = \mathrm{P}(Y_i \leqslant y|I), \quad i = 1, 2, \ldots \tag{6.1}$$

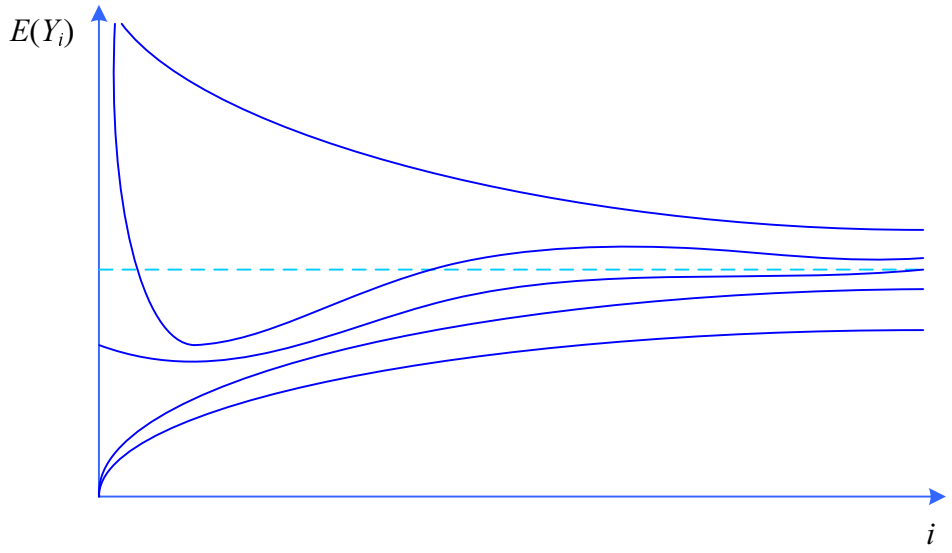
Na grafiku sa slike 6.2 mogu se uočiti dve oblasti, koje odgovaraju prelaznom i ustaljenom režimu. Odlika ustaljenog režima je da nastupa za dovoljno veliki broj replikacija i da u njemu izlaz simulacije ne zavisi od početnog uslova, tj. da važi

$$F_i(y|I) \to F(y), \quad (\forall I). \tag{6.2}$$

Iako ustaljeni režim ne zavisi od početnog uslova, brzina konvergencije ka njemu zavisi, što je ilustrovano na slici 6.3. Sa stanovišta dobijanja verodostojnih i reprezentativnih rezultata u što kraćem vremenu, poželjno je da izlaz simulacije što pre uđe u ustaljeni režim. Napomenimo da se u izlazu nekih simulacionih modela mogu javiti i oscilatorni procesi.

Da bi se eliminisao uticaj početnog uslova na ocenu parametara sistema, potrebno je iz rezultata izbaciti onaj deo koji se odnosi na prelazni režim, a koji se u literaturi naziva *warmup* ili *startup* periodom. U operativnom smislu, to npr. znači da ćemo uzorak izlaznog signala za statističku obradu izabrati tako da ne obuhvati l prvih odbiraka, koji odgovaraju prelaznom režimu, ili da ćemo iz razmatranja izostaviti l prvih korisnika koji dolaze u servisni sistem.

Praktičan postupak za identifikaciju početka ustaljenog režima zasniva se na Welchovoj grafičkoj proceduri. Po njoj, prvo se izvrši $n \geqslant 5$ replika simulacije, od kojih je svaka



Slika 6.3: Primer zavisnosti očekivanja rezultata simulacije od broja replikacija i početnog uslova.

dužine m. Potom se konstruiše usrednjen proces,

$$\overline{Y}_i = \frac{1}{n}\sum_{j=1}^{n} Y_{ji}, \tag{6.3}$$

gde je Y_{ji} i-ta realizacija u j-toj replikaciji. Nije teško zaključiti da je varijansa ovakvog procesa

$$\mathrm{Var}\overline{Y} = \frac{1}{n}\,\mathrm{Var}Y. \tag{6.4}$$

Usrednjeni proces se potom prevodi u oblik

$$\overline{Y}_i(w) = \begin{cases} \dfrac{1}{2w+1}\displaystyle\sum_{s=-w}^{w} \overline{Y}_{i+s}, & i = w+1,\ldots,m+w \\ \dfrac{1}{2i-1}\displaystyle\sum_{s=-(i-1)}^{i-1} \overline{Y}_{i+s}, & i = 1,2,\ldots,w \end{cases}. \tag{6.5}$$

Pri tome je sa w označena širina prozorske funkcije, za koju važi $w \geqslant \left[\frac{m}{4}\right]$. Može se pokazati da ovakva obrada usrednjenog procesa ima prirodu niskofrekvencjskog filtriranja.

Dalje se nacrta grafik $\overline{Y}_i(w)$ za $i = 1,2,\ldots,m-w$ i sa njega izabere l (tj. početak ustaljenog režima) kao vrednost indeksa i posle koje se $\overline{Y}_i(w)$ malo menja.

Metodološki ispravan postupak bio bi da se statistička obrada ne izvršava na seriji rezultata koja je korištena za procenu vrednosti l, već da se sada izvede nova serija od n' replikacija, iz koje će se uzeti $m' - l$ realizacija za obradu.

6.2 Ocene parametara

Uvedimo, na početku, neke važne pojmove. Populacija ili osnovni skup je skup svih mogućih ishoda simulacije, dok uzorak čine ishodi koje smo zaista dobili izvršavanjem simulacije.

Broj elemenata uzorka naziva se njegovim obimom. Uzorak je reprezentativan ako ima iste statističke osobine kao i populacija iz koje je uzet. Cilj metodološki dobre simulacije je da se reprezentativnost postigne već za uzorke maloga obima, mnogo manjeg od obima populacije; na taj način, smanjuje se potreban broj ponavljanja eksperimenta i brže se dolazi do verodostojnih rezultata.

Slučajna promenljiva koja je funkcija samo elemenata uzorka naziva se statistikom. Od naročitog su interesa statistike koje predstavljaju ocene parametara simuliranog sistema; ovakve statistike nazivaju se i estimatorima.

Označimo parametar koji analiziramo sa θ. Neka je $\hat{\theta}$ statistika koja se koristi kao njegova ocena na izabranom uzorku. Kaže se da je $\hat{\theta}$ centrirana ili nepristrasna ocena ako je

$$\mathrm{E}\hat{\theta} = \theta. \tag{6.6}$$

Ocena je stabilna ako je, za proizvoljno malo ϵ,

$$\lim_{n\to\infty} \mathrm{P}\left(|\hat{\theta} - \theta| > \epsilon\right) = 0. \tag{6.7}$$

Ako postoje dve statistike, bolja je ona čije je srednje kvadratno odstupanje od stvarne vrednosti (θ) manje.

Iako se u statistici to pitanje ne razmatra, u nastavku ćemo smatrati da je uzorak za ocenu parametara uzet iz ustaljenog režima izlaza simulacije.

6.2.1 Ocena matematičkog očekivanja

Neka je dat uzorak obima n, $X_1, X_2, \ldots, X_n$. Centrirana i stabilna ocena njegovog matematičkog očekivanja je

$$\hat{\mu} = \frac{X_1 + X_2 + \cdots + X_n}{n}. \tag{6.8}$$

6.2.2 Ocena varijanse

Statistike za ocenu varijanse se razlikuju u zavisnosti od toga da li se ocenjuje i matematičko očekivanje uzorka, ili je ono unapred poznato.

Ako matematičko očekivanje uzorka nije poznato, ocena varijanse data je izrazom

$$s^2 = \frac{1}{n-1} \sum_{k=1}^{n} (X_k - \hat{\mu})^2 . \tag{6.9}$$

Za praktične primene je pogodniji njegov sređeni oblik,

$$s^2 = \frac{1}{n-1} \sum_{k=1}^{n} X_k^2 - \frac{n}{n-1} \hat{\mu}^2. \tag{6.10}$$

Nije teško pokazati da je ova ocena centrirana i stabilna.

Kada je poznato matematičko očekivanje, koristi se ocena

$$s_0^2 = \frac{1}{n} \sum_{k=1}^{n} (X_k - \mu)^2 . \tag{6.11}$$

6.2.3 Ocena verovatnoće

Ocena verovatnoće se zasniva na statističkoj definiciji verovatnoće:

$$\hat{p} = \frac{\text{broj uspeha}}{\text{broj eksperimenata}}. \tag{6.12}$$

Na primer, ako ocenjujemo verovatnoću sudara u telekomunikacionom mreži, „broj uspeha" će biti broj sudarenih paketa, dok će „broj eksperimenata" biti ukupan broj generisanih paketa.

6.2.4 Ocena parametra Poissonove raspodele

Neka je $X_1, X_2, \ldots, X_n$ uzorak iz Poissonove raspodele. Parametar raspodele λ predstavlja njeno matematičko očekivanje, pa se ocenjuje po obrascu

$$\hat{\lambda} = \frac{X_1 + X_2 + \cdots + X_n}{n}. \tag{6.13}$$

6.2.5 Ocena srednje snage

Srednja snaga signala se ocenjuje po obrascu

$$\hat{P} = \frac{1}{n} \sum_{k=1}^{n} X_k^2. \tag{6.14}$$

6.2.6 Ocena spektralne gustine srednje snage

U frekvencijskom domenu, spektralna gustina srednje snage signala se ocenjuje na osnovu njegovog periodograma:

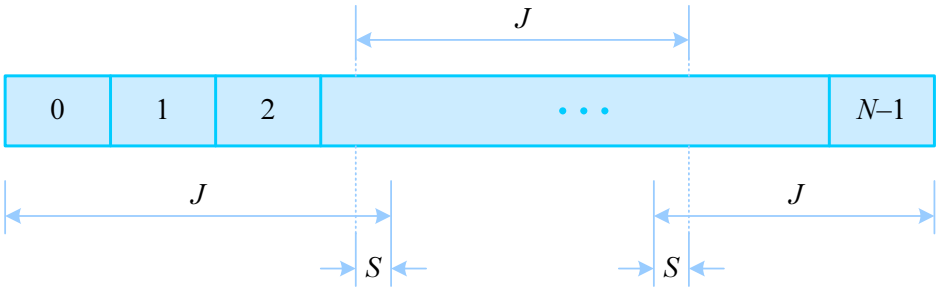
$$S_{XX}(f) = \lim_{T\to\infty} \mathrm{E}\,\frac{|X_T(f)|^2}{T}, \tag{6.15}$$

gde je sa T označeno trajanje uočenog segmenta signala.

Često korišten metod za ocenu spektralne gustine srednje snage zasniva se na tzv. Welchovom postupku, koji je opisan sledećim koracima:

- Uzme se N odbiraka signala $x(n)$, $n = 0, 1, \ldots, N-1$.
- Signal se izdeli na M segmenata od po J odbiraka, tako da se sukcesivni segmenti uzajamno preklapaju u S odbiraka.
- Za svaki segment se izračunaju DFT i periodogram.
- Spektralna gustina srednje snage signala se dobije usrednjavanjem dobijenih periodograma.

Podela signala na segmente je ilustrovana na slici 6.4.



Slika 6.4: Ocena spektralne gustine srednje snage signala Welchovim postupkom.

Nije teško uočiti da između parametara J, S i M postoji sledeća veza:

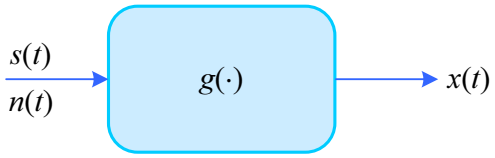
$$M = \left[\frac{N-J}{S+1}\right]. \tag{6.16}$$

Welch je preporučio da preklapanje segmenata iznosi 50%, kao i da se pre izračunavanja DFT signal uobliči Hannovom prozorskom funkcijom.

Napomenimo da se spektralna gustina srednje snage signala može ocenjivati i u vremenskom domenu, korišćenjem autokorelacionih funkcija.

6.2.7 Ocena odnosa signal-šum

Posmatrajmo nelinearan sistem s memorijom, na čiji ulaz dolaze signal $s(t)$ i slučajni šum $n(t)$. Označimo slučajni proces na izlazu sistema sa $x(t)$.



Slika 6.5: Nelinearni sistem s memorijom.

Postupak ocene je sledeći:

- Simulacijom se odredi M odbiraka signala na izlazu, $x(k)$, $k = 1, 2, \ldots, M$ i $N = M + K$ odbiraka signala na ulazu, $s(k)$, $k = -(K-1), -(K-2), \ldots, 0, 1, \ldots, M$. K dodatnih odbiraka treba da „pokriju" maksimalno očekivano kašnjenje signala prilikom prenosa kroz posmatrani sistem.
- Sekvenca x se dopuni s K nula.
- Izračunaju se diskretne Fourierove transformacije sekvenci x i s.
- Izračuna se inverzna diskretna Fourierova transformacija proizvoda SX^*.
- Nađe se indeks koeficijenta, k, čija je amplituda najveća; za to je potrebno pretražiti prvih $K+1$ koeficijenata. Proračuna se vremensko kašnjenje τ koje odgovara ovome indeksu.

Kada je poznato τ, odnos signal-šum se ocenjuje po obrascu

$$\hat{\rho} = \frac{R_{xs}^2(\tau)}{\langle X^2 \rangle \langle s^2(t-\tau) \rangle - R_{xs}^2(\tau)}, \tag{6.17}$$

gde je sa R_{xs} označena uzajamna korelacija signala x i s, dok je $\langle \cdot \rangle$ operator usrednjavanja.

6.2.8 Očekivanje i varijansa Monte Carlo estimatora

Neka se serija Monte Carlo simulacija sastojala od izvođenja n Bernoullijevih eksperimenata; na primer, generisano je n bita, koji su preneseni nelinearnim kanalom u kome deluje šum, pa je na prijemu ocenjena verovatnoća greške:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} e_i, \tag{6.18}$$

gde je e_i indikatorska promenljiva,

$$e_i = \begin{cases} 0, & \text{ako je } i\text{-ti bit prenesen ispravno} \\ 1, & \text{ako je } i\text{-ti bit prenesen pogrešno} \end{cases}. \tag{6.19}$$

Očekivanje ovoga estimatora je

$$\mathrm{E}\hat{p} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}e_i = \frac{1}{n}\, n\, \mathrm{E}e_i = \mathrm{E}e_i = p, \tag{6.20}$$

pa je Monte Carlo estimator centriran ili nepristrasan.

Uz pretpostavku da su greške uzajamno nezavisne, varijansa Monte Carlo estimatora je

$$\begin{aligned}\sigma^2 = \mathrm{E}\hat{p}^2 - (\mathrm{E}\hat{p})^2 &= \mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n} e_i\right)^2 - p^2 = \\ &= \frac{1}{n^2}\,\mathrm{E}\left(\sum_{i=1}^{n} e_i^2 + \sum_{\substack{i,j=1,\ldots,n \\ i\neq j}}\sum e_i e_j\right) - p^2 = \\ &= \frac{1}{n^2}\left(np + (n^2-n)p^2\right) - p^2 = \\ &= \frac{p\,(1-p)}{n}. \end{aligned} \tag{6.21}$$

Vidimo da varijansa Monte Carlo estimatora linearno opada s porastom obima uzorka.

U praksi se definiše i tzv. normalizovana standardna greška ili relativna preciznost:

$$\epsilon = \frac{\sigma}{p} = \sqrt{\frac{1-p}{np}} \approx \frac{1}{\sqrt{np}}, \tag{6.22}$$

što opada s kvadratnim korenom broja replikacija. Odavde se, za zadato ϵ, može proceniti potreban broj replikacija simulacije kao

$$n = \frac{1}{\epsilon^2 p}. \tag{6.23}$$

6.2.9 Ocene parametara raspodele u programu GNU Octave

Pokazaćemo kako se u programu GNU Octave iz datog uzorka ocenjuju parametri raspodele. Učitaćemo uzorak iz datoteke, nacrtaćemo njegov histogram, ocenićemo matematičko očekivanje i varijansu i, na kraju, sačuvaćemo rezultate u izlaznoj datoteci.

Proverićemo prvo koji je radni direktorijum:

```
> pwd
ans = C:\Program Files\Octave
```

Postavićemo da radni direktorijum bude onaj u kome je datoteka sa uzorkom:

```
> cd D:\OT4MIS
```

Ako putanja do direktorijuma sadrži razmake, treba je pisati među navodnicima; u tom slučaju, umesto simbola `\` treba kucati `/`.

Učitaćemo datoteku s podacima:

```
> X = load("uzorak.txt");
```

U nastavku ćemo smatrati da je X vektor.

Histogram uzorka s k stubića dobićemo naredbom

```
> hist(X, k)
```

Matematičko očekivanje i varijansu ocenićemo naredbama

```
> mu = mean(X)
> s2 = var(X)
```

Prilikom ocene varijanse, smatramo da matematičko očekivanje *nije poznato*, pa koristimo formulu (6.9):

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu})^2 ,$$

gde je $\hat{\mu}$ ocena matematičkog očekivanja. Ukoliko bi očekivanje bilo poznato, koristili bismo naredbu `var(X, 1)`, koja bi računala varijansu s težinskim faktorom $1/n$.

Rezultate izračunavanja (npr. neku promenljivu Y) sačuvaćemo u tekstualnoj datoteci *izlaz* naredbom

```
> save("-ascii", "izlaz.txt", "Y")
```

6.3 Intervali poverenja

Interval poverenja za parametar θ s nivoom poverenja $1-\alpha$ je interval $[Y_1, Y_2]$, za koji važi

$$\mathrm{P}\left(\theta \in [Y_1, Y_2]\right) = 1 - \alpha. \tag{6.24}$$

6.3.1 Intervali poverenja za matematičko očekivanje

Označimo sa ϵ_u kvantil reda u iz standardne normalne raspodele[1]. Pri određivanju intervala poverenja za matematičko očekivanje treba razlikovati slučajeve kada je varijansa poznata i kada nije.

Ako je varijansa poznata, tada je dvostrani interval poverenja za matematičko očekivanje

$$\mathrm{P}\left(\mu \in \left[\hat{\mu} - \epsilon_{1-\alpha/2}\frac{\sigma}{\sqrt{n}},\ \hat{\mu} + \epsilon_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}\right]\right) = 1 - \alpha, \tag{6.25}$$

[1] $\Phi(\epsilon_u) = u$

dok su jednostrani intervali poverenja

$$\mathrm{P}\left(\mu \in \left(-\infty,\ \hat{\mu} + \epsilon_{1-\alpha}\frac{\sigma}{\sqrt{n}}\right]\right) = 1-\alpha, \tag{6.26}$$

$$\mathrm{P}\left(\mu \in \left[\hat{\mu} - \epsilon_{1-\alpha}\frac{\sigma}{\sqrt{n}},\ +\infty\right)\right) = 1-\alpha. \tag{6.27}$$

Ovi intervali su izvedeni za nezavisan uzorak iz raspodele $\mathcal{N}(\mu, \sigma^2)$, pri čemu je σ^2 poznato. Ako je uzorak velikog obima, što se u praksi postiže za $n \geqslant 30$, ovo su dobre aproksimacije i za raspodele koje nisu normalne.

Kada σ^2 nije poznato, u gornjim izrazima umesto tačne vrednosti standardne devijacije, σ, figuriše njena ocena, s. ϵ_u je sada kvantil reda u iz Studentove raspodele sa $n-1$ stepenom slobode, $t(n-1)$; za velike uzorke ($n \geqslant 30$), ova raspodela se svodi na normalnu.

6.3.2 Intervali poverenja za varijansu

Ukoliko je uzorak uzet iz normalne raspodele, intervali poverenja za varijansu dati su izrazima

$$\mathrm{P}\left(\sigma^2 \in \left[\frac{(n-1)s^2}{\epsilon_{1-\alpha/2}},\ \frac{(n-1)s^2}{\epsilon_{\alpha/2}}\right]\right) = 1-\alpha, \tag{6.28}$$

$$\mathrm{P}\left(\sigma^2 \in \left[0,\ \frac{(n-1)s^2}{\epsilon_{\alpha}}\right]\right) = 1-\alpha, \tag{6.29}$$

$$\mathrm{P}\left(\sigma^2 \in \left[\frac{(n-1)s^2}{\epsilon_{1-\alpha}},\ +\infty\right)\right) = 1-\alpha, \tag{6.30}$$

gde je ϵ_u kvantil reda u raspodele $\chi^2(n-1)$.

6.3.3 Intervali poverenja za ocenu verovatnoće

Sledeći intervali se dobijaju za uzorke velikog obima ($n \geqslant 30$), primenom centralne granične teoreme:

$$\mathrm{P}\left(p \in \left[\hat{p} - \epsilon_{1-\alpha/2}\frac{1}{2\sqrt{n}},\ \hat{p} + \epsilon_{1-\alpha/2}\frac{1}{2\sqrt{n}}\right]\right) = 1-\alpha, \tag{6.31}$$

$$\mathrm{P}\left(p \in \left[0,\ \hat{p} + \epsilon_{1-\alpha}\frac{1}{2\sqrt{n}}\right]\right) = 1-\alpha, \tag{6.32}$$

$$\mathrm{P}\left(p \in \left[\hat{p} - \epsilon_{1-\alpha}\frac{1}{2\sqrt{n}},\ 1\right]\right) = 1-\alpha. \tag{6.33}$$

U gornjim izrazima, ϵ_u je kvantil reda u iz standardne normalne raspodele $\mathcal{N}(0,1)$.

6.3.4 Intervali poverenja za parametar Poissonove raspodele

Neka je ϵ_u je kvantil reda u iz standardne normalne raspodele $\mathcal{N}(0,1)$. Intervali poverenja za parametar Poissonove raspodele λ su

$$\mathrm{P}\left(\lambda \in \left(\hat{\lambda} - \epsilon_{1-\alpha/2}\sqrt{\frac{\hat{\lambda}}{n}},\ \hat{\lambda} + \epsilon_{1-\alpha/2}\sqrt{\frac{\hat{\lambda}}{n}}\right)\right) = 1-\alpha, \tag{6.34}$$

$$\mathrm{P}\left(\lambda \in \left(0,\ \hat{\lambda} + \epsilon_{1-\alpha}\sqrt{\frac{\hat{\lambda}}{n}}\right)\right) = 1-\alpha, \tag{6.35}$$

$$\mathrm{P}\left(\lambda \in \left(\hat{\lambda} - \epsilon_{1-\alpha}\sqrt{\frac{\hat{\lambda}}{n}},\ +\infty\right)\right) = 1-\alpha. \tag{6.36}$$

6.3.5 Intervali poverenja u programu GNU Octave

Za određivanje intervala poverenja, potrebni su nam kvantili normalne, χ^2 i Studentove t-raspodele. Dobićemo ih komandama

```
> e1 = norminv (u, m, s)
> e2 = chi2inv(u, d)
> e3 = tinv(u, d)
```

gde je u red kvantila, m matematičko očekivanje, s standardna devijacija i d broj stepeni slobode raspodele. Ukoliko se radi o standardnoj raspodeli $\mathcal{N}(0,1)$, argumenti m i s mogu se izostaviti, jer su njhove podrazumevane vrednosti 0 i 1, respektivno.

Standardnu devijaciju, koja figuriše u formulama za izračunavanje granica intervala, možemo izračunati bilo kao kvadratni koren varijanse, bilo naredbom `std`.

6.4 Praktične napomene

U praktičnim primenama, najčešće se ocenjuje vrednost matematičkog očekivanja. Ocena se po pravilu ne izvodi iz jednog uzorka, već se koriste postupci koje ćemo objasniti u nastavku.

S metodom nezavisnih replikacija smo se upoznali u poglavlju o Monte Carlo simulaciji (strana 59). Ovde se izvršava n replikacija simulacije, od kojih je svaka obima k; pod obimom ovde podrazumevamo broj statističkih eksperimenata, npr. broj prenesenih bita, tačaka u kojima se izračunava vrednost integrala, vremenskih intervala u kojima se mogu dešavati događaji i slično. Za svaku replikaciju se, na ustaljenom režimu,

oceni vrednost očekivanja, čime se dobija n nezavisnih i identično raspodeljenih ocena. Za ovako izvedeni uzorak se potom ocenjuje matematičko očekivanje i odredi interval poverenja takve ocene. Dobra strana ovoga pristupa je u tome što je koncepcijski jednostavan, a loša je u tome što traži veliki broj ponavljanja nezavisnih statističkih eksperimenata ($n \cdot k$).

Metoda srednje vrednosti nizova se u praksi najčešće koristi. Ona se zasniva na činjenici da su (vremenski) dovoljno udaljeni elementi izlaznog niza međusobno slabo korelisani. Ovde se izvršava jedna replikacija obima k. Iz izlaza se odbaci deo koji se odnosi na prelazni režim i ostatak se izdeli na podnizove dužine m, $m \ll k$, tako da odgovarajući članovi različitih podnizova budu uzajamno praktično nezavisni. Za svaki podniz se oceni matematičko očekivanje; ove ocene će biti identično raspodeljene, jer se odnose na ustaljeni režim, ali takođe i nezavisne, jer su podnizovi uzajamno dovoljno razmaknuti. Slično kao i u prethodnom pristupu, za ovaj novi uzorak se oceni očekivanje i odredi interval poverenja ocene.

Cilj treće, regenerativne metode je pronaći trenutak u ustaljenom režimu u kome se sistem vraća u početno stanje (tj. u kome se stanje sistema „resetuje"); ovaj trenutak se naziva tačkom regeneracije. Na primer, tačka regeneracije za servisni sistem $G/G/1$ je svaki trenutak u kome korisnik pri dolasku zatiče prazan sistem. Odzivi sistema između dvaju sukcesivnih tačaka regeneracije se tretiraju kao nezavisni nizovi na kojima se, kao u prethodnim metodima, ocenjuje očekivanje. Ova metoda je konceptualno najelegantnija, ali i najteža za implementiranje, jer je u praksi često teško odrediti tačku regeneracije, a može se desiti i da su one međusobno isuviše udaljene.

6.5 Statističke granice

Navešćemo, bez izvođenja, dve važne statističke granice.

Nejednakost Chebysheva se navodi u dva oblika:

$$\mathrm{P}\left(|X| \geqslant \epsilon\right) \leqslant \frac{1}{\epsilon^2}\,\mathrm{E}\left(X^2\right), \tag{6.37}$$

$$\mathrm{P}\left(|Y-\mu| \geqslant k\sigma\right) \leqslant \frac{1}{k^2}. \tag{6.38}$$

Nejednakost Chernoffa data je izrazom

$$\mathrm{P}\left(X \geqslant \epsilon\right) \leqslant \min_{t \geqslant 0} e^{-t\epsilon}\,\mathrm{E}\left(e^{tX}\right). \tag{6.39}$$

Preciznija je od nejednakosti Chebysheva, ali u opštem slučaju nije pogodna za primene, jer zahteva izračunavanje člana $\mathrm{E}\left(e^{tX}\right)$.

Ukoliko uzorak potiče iz raspodele $\mathcal{N}(0,1)$, nejednakost Chernoffa se svodi na jednostavniji oblik

$$\mathrm{P}\left(X \geqslant \epsilon\right) \leqslant e^{-\epsilon^2/2}. \tag{6.40}$$

6.6 Testovi hipoteza

Dokazivanje u statistici nije egzaktno, već se izvodi testiranjem hipoteza. Metodološki ispravan postupak za dokazivanje nekog tvrđenja bio bi sledeći: Suprotno tvrđenje (ili neutralno ili postojeće stanje) uzimamo za tzv. nultu hipotezu, H_0, dok samo tvrđenje koje dokazujemo uzimamo za hipotezu H_1. Cilj statističkog testa je ispitati ima li dokaza protiv H_0, u korist H_1; da bi to bilo moguće, treba definisati statistiku testa, X i skup njenih vrednosti za koje se odbacuje nulta hipoteza, tzv. oblast odbacivanja. Moguća su dva ishoda testa:

- Ako je vrednost statistike testa unutar oblasti odbacivanja, odbacujemo hipotezu H_0, a prihvatamo H_1.
- Ako je vrednost statistike testa van oblasti odbacivanja, nema dokaza protiv hipoteze H_0, pa je ne odbacujemo.

Po analogiji s teorijom odlučivanja iz telekomunikacija, vidimo da su pri testiranju hipoteza moguće dve vrste greške. Greška prve vrste nastupa kada se hipoteza H_0 odbaci, iako je tačna; greška druge vrste nastupa onda kada se H_0 ne odbaci, iako je tačna hipoteza H_1.

Maksimalna vrednost greške prve vrste naziva se nivoom značajnosti testa. Uobičajeno se koriste sledeće vrednosti nivoa značajnosti: 0,1; 0,05; 0,025 i 0,01. Što je manji nivo značajnosti, utoliko je za isti obim uzorka i za istu statistiku testa teže odbaciti hipotezu H_0. Smanjivanjem nivoa značajnosti, povećava se verovatnoća greške druge vrste.

Testovi u kojima se hipoteze odnose na vrednosti parametara raspodele nazivaju se parametarskim testovima, dok se u neparametarskim testovima hipoteze ne odnose na vrednosti parametara, već npr. na to da li posmatrani uzorak potiče iz zadate raspodele, ili ne.

Na kraju ovoga uvoda, osvrnimo se još jednom na izbor hipoteza. Rekli smo da bi bilo poželjno da H_0 bude tvrđenje koje opovrgavamo, a da H_1 bude tvrđenje koje dokazujemo. Sa stanovišta formalnog sprovođenja testa, neophodno je da poznajemo raspodelu statistike u slučaju kad je u važnosti nulta hipoteza; hipoteze ćemo stoga birati tako da zadovoljimo ovaj uslov – kao nultu ćemo birati onu hipotezu pod čijim je važenjem poznata statistika testa, bez obzra na to koje stanje ona odražava.

6.6.1 Pearsonov (χ^2) test

Pearsonov ili χ^2 (hi kvadrat) test spada u najčešće korišćene statističke testove. Kao što ćemo videti, on se svodi na poređenje empirijskog histograma s pretpostavljenim.

Testiranje hipoteza o raspodeli

Neka je $(X_1, X_2, \ldots, X_n)$ nezavisan uzorak iz nepoznate raspodele. Cilj testa je da utvrdimo da li se empirijska funkcija raspodele uzorka, F, podudara s nekom zadatom funkcijom raspodele, F_0.

Izdelimo realnu osu na r disjunktnih intervala $A_j = (a_{j-1}, a_j]$, $j = 1, 2, \ldots, r$, pri čemu je $a_0 = -\infty$, $a_r = +\infty$, kao što je to prikazano na slici 6.6.

Slika 6.6: Podela realne ose na intervale.

Primetimo da ovi intervali (koji se nazivaju i klasama) u opštem slučaju nisu ekvidistantni.

Neka je X_i i-ti element uzorka. Verovatnoća da se on nalazi u intervalu A_j je

$$p_j = \mathrm{P}\left(X_i \in A_j\right) = F(a_j) - F(a_{j-1}). \tag{6.41}$$

Kada bi važila pretpostavljena raspodela, verovatnoća nalaženja ishoda u ovom intervalu bila bi

$$p_{j0} = F_0(a_j) - F_0(a_{j-1}). \tag{6.42}$$

Hipoteze koje testiramo su

$$H_0: \quad p_1 = p_{10}, p_2 = p_{20}, \ldots, p_r = p_{r0},$$

$$H_1: \quad (p_1, p_2, \ldots, p_r) \neq (p_{10}, p_{20}, \ldots, p_{r0}).$$

Imajući u vidu sliku 6.6, nije teško zaključiti da ovim zaista testiramo podudaranje empirijskog histograma s pretpostavljenim.

Neka je, dalje, n obim uzorka i N_j broj elemenata uzorka koji se nalaze u j-tom intervalu. Statistika testa data je izrazom

$$\chi^2 = \sum_{j=1}^{r} \frac{(N_j - np_{j0})^2}{np_{j0}} \tag{6.43}$$

i asimptotski (za veliko n) ima hi kvadrat raspodelu sa $r - 1$ stepenom slobode.

Zadržimo se na izrazu za statistiku testa. N_j je stvarni broj ishoda, a np_{j0} očekivani, kada bi važila nulta hipoteza. Statistika testa stoga ima značenje

$$\chi^2 = \sum \frac{(\text{stvarno} - \text{očekivano})^2}{\text{očekivano}}.$$

Nulta hipoteza se odbacuje s nivoom značajnosti α ako i samo ako je $\chi^2 > \epsilon_{1-\alpha}$, gde je $\epsilon_{1-\alpha}$ kvantil reda $1 - \alpha$ raspodele $\chi^2(r-1)$.

Pri praktičnim primenama ovoga testa, intervale treba izabrati tako da očekivani broj ishoda u svakom od njih bude barem 5. Poželjno je i da ne bude velikih razlika u očekivanom broju ishoda po intervalima.

Prethodno izlaganje se odnosi na slučaj kada su poznati parametri pretpostavljene raspodele; test se može primeniti i kada oni nisu poznati, već se ocenjuju iz uzorka. Ako se metodom maksimalne verodostojnosti oceni k parametara, statistika testa sada će pod hipotezom H_0 asimptotski imati raspodelu $\chi^2(r-1-k)$.

Testiranje saglasnosti podataka s pretpostavljenim modelom

Neka je izvršeno n nezavisnih eksperimenata sa po r ishoda. Neka je, dalje, verovatnoća j-tog ishoda p_j i broj realizacija j-tog ishoda u seriji od n eksperimenata N_j.

„Model" je pretpostavljena raspodela uzorka, čiji je parametar $\boldsymbol{\theta}$ vektor dimenzije k; ukoliko nulta hipoteza ne zavisi od parametra, formalno usvajamo $k = 0$. Statistika testa je

$$\chi^2 = \sum_{j=1}^{r} \frac{(N_j - np_{j0}(\hat{\boldsymbol{\theta}}))^2}{np_{j0}(\hat{\boldsymbol{\theta}})} \tag{6.44}$$

i pod nultom hipotezom, asimptotski ima raspodelu $\chi^2(r-1-k)$.

Kritična vrednost testa, c, s nivoom značajnosti α je kvantil reda $1-\alpha$ raspodele $\chi^2(r-1-k)$. Nulta hipoteza se odbacuje ako i samo ako je $\chi^2 > c$.

Testiranje nezavisnosti

Neka se u svakom od n eksperimenata istovremeno realizuju dva ishoda. Označimo sa f_{ij} broj eksperimenata u kojima su se realizovali ishodi A_i i B_j, $i = 1, \ldots, v$, $j = 1, \ldots, k$. Ovaj podatak možemo pregledno prikazati tablicom kontingencije.

Tabela 6.1: Tablica kontingencije.

	B_1	B_2	$\cdots$	B_k	ukupno
A_1	f_{11}	f_{12}	$\cdots$	f_{1k}	a_1
A_2	f_{21}	f_{22}	$\cdots$	f_{2k}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
A_v	f_{v1}	f_{v2}	$\cdots$	f_{vk}	a_v
ukupno	b_1	b_2	$\cdots$	b_k	n

Statistika testa je

$$\chi^2 = \sum_{i=1}^{v}\sum_{j=1}^{k} \frac{(nf_{ij} - a_i b_j)^2}{na_i b_j}. \tag{6.45}$$

Hipoteza H_0 o nezavisnosti događaja A_i i B_j se odbacuje s nivoom značajnosti α ako i samo ako je statistika testa veća od kvantila reda $1-\alpha$ raspodele $\chi^2((v-1)(k-1))$.

Testiranje generatora slučajnih brojeva

Hipoteza H_0 pretpostavlja pripadnost uzorka slučajnih brojeva obima n zadatoj raspodeli, na primer uniformnoj. Ukoliko važi H_0, χ^2 statistika ima raspodelu $\chi^2(n-1)$. Ako su brojevi iz uzorka zaista slučajni, nisu poželjne ni isuviše velike, niti isuviše male vrednosti statistike testa; prve bi, npr. odgovarale binarnoj sekvenci 11111..., a druge sekvenci 10101... Oblast odbacivanja nulte hipoteze stoga se modifikuje u odnosu na osnovnu formulaciju testa i sada je $\chi^2 < \epsilon_{\alpha/2} \vee \chi^2 > \epsilon_{1-\alpha/2}$.

6.6.2 Test Kolmogorova i Smirnova

Dok hi kvadrat test poredi histograme, test Kolmogorova i Smirnova (K-S) poredi funkcije raspodele.

Neka je F_n empirijska funkcija raspodele uzorka obima n. Testom ispitujemo poklapanje ove raspodele s pretpostavljenom, F_0. Hipoteze su, stoga, sledećeg oblika:

$$H_0: \quad F = F_0,$$

$$H_1: \quad F \neq F_0.$$

Statistika testa data je izrazom

$$\lambda = \sqrt{n} \sup_{x \in \mathbb{R}} |F_n(x) - F_0(x)| \tag{6.46}$$

i za uzorak velikog obima ima K-raspodelu Kolmogorova.

Hipoteza H_0 se odbacije s nivoom značajnosti α ako je $\lambda > c$, gde je c kvantil reda $1-\alpha$ K-raspodele.

6.6.3 Testovi hipoteza u programu GNU Octave

Pokazaćemo kako se u programu GNU Octave izvršavaju hi kvadrat test i test Kolmogorova i Smirnova.

Testiranje hipoteza o raspodeli χ^2 testom

Primenom *sume 12*, generisali smo slučajni vektor Z sa 100 elemenata. Testiraćemo hipotezu da oni doista imaju normalnu raspodelu.

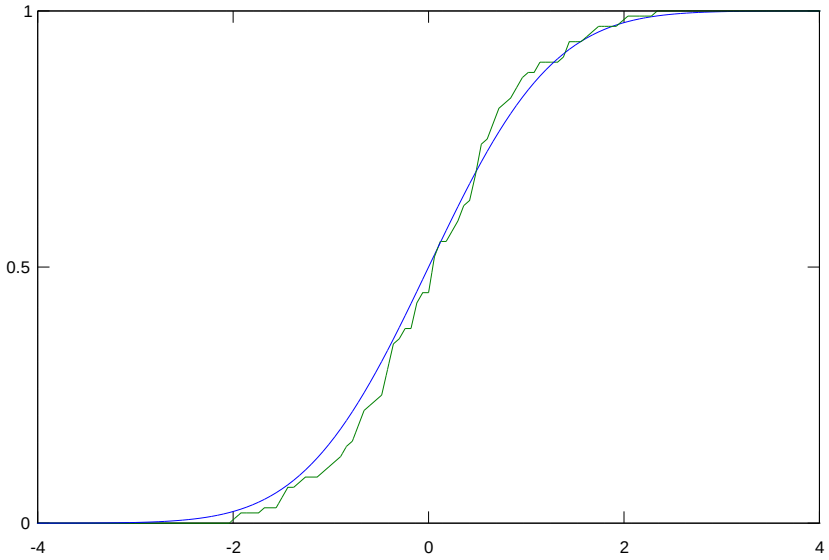
Generisaćemo i slučajni uzorak od 100 brojeva iz $\mathcal{N}(0,1)$:

```
> C = normrnd(0, 1, 1, 100);
```

Metoda *sume 12* daje slučajne brojeve koji pripadaju intervalu $[-6,6]$. Generišimo pomoćni vektor X i empirijsku i pretpostavljenu funkciju raspodele:

```
> X = (-1:0.01:1)*6;
> E = empirical_cdf(X, Z);
> N = normcdf(X, 0, 1);
```

Grafici empirijske funkcije raspodele (dobijene na uzorku) i pretpostavljene funkcije raspodele prikazani su na slici 6.7. Grafici su dobijeni naredbama `plot(X, E, X, N)`, `axis([-4 4 0 1])`.



Slika 6.7: Empirijska i pretpostavljena funkcija raspodele.

Hipotezu H_0 da naš uzorak pripada normalnoj raspodeli testiramo naredbom

```
> [alpha, chisq, d] = chisquare_test_homogeneity(Z, C, (-3:0.5:3))
```

Ulazni argumenti su, redom, sekvenca koju testiramo, sekvenca iz pretpostavljene raspodele i granice intervala (klasa). U gornjem primeru smo, bez mnogo razmišljanja, zadali intervale $(-\infty,-3]$, $(-3,-2{,}5]$, $(-2{,}5,-2]$, $\ldots$, $(2{,}5,3]$ i $(3,\infty]$.

Izlazi testa su, redom, granična vrednost nivoa značajnosti pri kojoj, za testirani uzorak, važi hipoteza H_0, vrednost Pearsonove (χ^2) statistike testiranog uzorka i broj stepeni slobode χ^2 raspodele.

Rezultati koje dobijamo su: $\alpha = 0{,}93377$, $\chi^2 = 6{,}3191$ i $d = 13$. Sada treba uporediti dobijeno α sa zadatim nivoom značajnosti testa. Ako je proračunata vrednost manja od zadate, odbacujemo hipotezu H_0.

Neka je u našem slučaju zahtevani nivo značajnosti 0,01. Dobijeno α je daleko veće, pa hipotezu H_0 *ne odbacujemo*. Primetimo da je vrednost χ^2 statistike mala, odnosno

da je granična vrednost α velika, pa izgleda da imamo kvalitetan uzorak iz normalne raspodele.

Test Kolmogorova i Smirnova

Test Kolmogorova i Smirnova se izvršava sledećom naredbom:

```
> [alpha, lambda] = kolmogorov_smirnov_test(Z, "normal", 0, 1)
```

Ulazni argumenti su, redom, testirani uzorak, pretpostavljena raspodela i njeni parametri. Izlazi su, redom, granična vrednost nivoa značajnosti pri kojoj, za testirani uzorak, važi hipoteza H_0 i vrednost K-S statistike uzorka.

U našem primeru, dobili smo $\alpha = 0{,}63485$ i $\lambda = 0{,}74560$. Ponovo na nivou značajnosti 0,01 nemamo dokaza protiv H_0.

Testiranje nezavisnosti χ^2 testom

Tablicu kontingencije generišemo naredbom `table`, čiji su argumenti vektori realizacije ishoda složenog eksperimenta.

Neka složeni eksperiment obuhvata istovremeno bacanje novčića (0 – pismo, 1 – glava) i kocke. Neka su u 10 eksperimenata registrovani sledeći ishodi: (0, 5), (0, 2), (1, 3), (0, 6), (1, 2), (1, 4), (1, 3), (0, 1), (0, 1), i (0, 5). Tablicu kontingencije generišemo na sledeći način:

```
> A = [0 0 1 0 1 1 1 0 0 0];
> B = [5 2 3 6 2 4 3 1 1 5];
> [T, La, Lb] = table(A, B)
```

Izlazi su, redom, frekvencije događaja A_iB_j i nivoi (liste) događaja A_i i B_j. Ako se funkcija pozove samo s jednim izlaznim argumentom, vratiće samo matricu frekvencija, T.

Kada je poznata tablica kontingencije, hipotezu H_0 o nezavisnosti događaja A_i i B_j testiramo naredbom

```
[alpha, chisq, d] = chisquare_test_independence(T)
```

Njen ulazni argument je tablica (matrica) kontingencije, dok je značenje izlaznih veličina isto kao kod χ^2 testa.

Ilustrujmo ovaj test na primeru 190 iz udžbenika prof. Merklea. 500 stanovnika nekog mesta u Americi odgovaralo je na pitanja o političkoj pripadnosti i o stavu prema novom programu štednje energije. Rezultati su prikazani u tablici kontingencije. Potrebno je

Tabela 6.2: Rezultati ispitivanja javnog mnenja.

	Za	Nema mišljenje	Protiv	Ukupno
Demokrati	138	83	64	285
Republikanci	64	67	84	215
Ukupno	202	150	148	500

s nivoom značajnosti 5% testirati hipotezu da je stav prema programu nezavisan od političke pripadnosti.

Pošto je poznata tablica kontingencije, nema potrebe za pozivanjem funkcije `table`, već ćemo odmah kucati

```
> T = [138 83 64; 64 67 84];
> [alpha, chisq, d] = chisquare_test_independence(T)
```

Rezultati su $\alpha = 1{,}5476 \cdot 10^{-5}$, $\chi^2 = 22{,}152$ i $d = 2$. Dobijena granična vrednost za α daleko je manja od usvojenog nivoa značajnosti, pa odbacujemo hipotezu o nezavisnosti.

6.7 Dodaci

6.7.1 Ocene kvantila u programu GNU Octave

Kvantil reda p iz raspodele $\mathrm{F}(x)$ je x_p za koje važi

$$\mathrm{F}(x_p) = p. \tag{6.47}$$

Kvantil reda 0,5 naziva se medijanom. Kvantili reda 0,25 i 0,75 nazivaju se, redom, prvim i trećim kvartilom. Kvantili reda n %, $0 \leqslant n \leqslant 100$ nazivaju se percentilima. Medijana je drugi kvartil ili pedeseti percentil.

Kvantile na uzorku ocenjujemo tako što konstruišemo empirijsku funkciju raspodele i potom iz nje nađemo x_p za koje je zadovoljen izraz (6.47). Za neke vrednosti p i neke uzorke, moći ćemo očitati tačnu vrednost, dok ćemo drugde morati da je procenimo između dvaju najbližih ishoda.

U nastavku je dat programski kod u GNU Octave kojim se ocenjuju kvantili.

```
function  q = quantile(x,p)
% ocena kvantila reda p na uzorku x
x = sort(x);
p = p(:);
n = size(x, 1);
x = [x(1, :); x; x(n, :)];
```

```
i = p*n + 1.5;
iu = ceil(i);
il = floor(i);
d = (i-il)*ones(1, size(x, 2));
qq = x(il, :).*(1-d) + x(iu, :).*d;
q(:) = qq;
endfunction
```

Naredbom `statistics` se redom računaju minimalna vrednost, prvi kvartil, medijana, treći kvartil, maksimalna vrednost, standardna devijacija, koeficijent asimetrije (*skewness*) i koeficijent spljoštenosti (*kurtosis*) uzorka.

6.7.2 Ocene parametara u Pythonu

U programu Python, obrada rezultata simulacije i ocena parametara su podržani bibliotekama SciPy i NumPy. Parametri uzorka se ocenjuju na sledeći način:

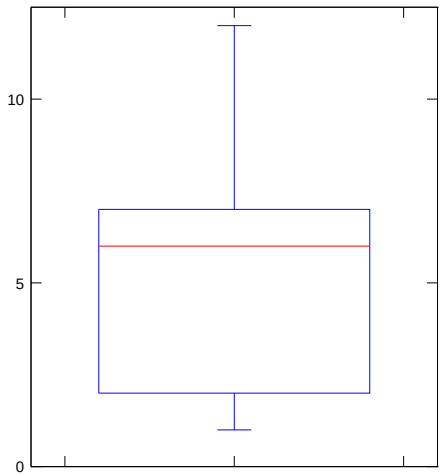
```
min(X)         # minimalna vrednost
max(X)         # maksimalna vrednost
mean(X)        # matematičko očekivanje
median(X)      # medijana
var(X)         # varijansa (u imeniocu N)
var(X, ddof=1) # varijansa (u imeniocu N-1)
std(X)         # standardna devijacija (u imeniocu N)
std(X, ddof=1) # standardna devijacija (u imeniocu N-1).
```

Kvantile raspodela ćemo izračunati u modulu `scipy.stats`:

```
from scipy import stats

# normalna raspodela
stats.norm.ppf(u, loc = m, scale = s) # u - red kvantila
                                      # m - matematičko očekivanje
                                      # s - standardna devijacija

# hi kvadrat raspodela
stats.chi2.ppf(u, d, loc = 0, scale = 1) # u - red kvantila
                                         # d - broj stepeni slobode

# t-raspodela
stats.t.ppf(u, d, loc = 0, scale = 1) # u - red kvantila
                                      # d - broj stepeni slobode.
```

6.7.3 Boks-dijagrami

Boks-dijagram je grafik na kome su redom prikazane minimalna vrednost uzorka, prvi kvartil, medijana, treći kvartil i maksimalna vrednost. Ove vrednosti su predstavljene horizontalnim linijama, kao što je to prikazano na slici 6.8.



Slika 6.8: Primer boks-dijagrama.

Boks-dijagrami se i u programu GNU Octave i u Pythonu (biblioteka matplotlib) crtaju naredbom `boxplot`. U najjednostavnijem obliku, njen jedini argument je uzorak, koji u GNU Octave mora biti u vidu vektora-kolone. Naredba crta boks-dijagram, a u GNU Octave vraća i numeričke vrednosti, koje redom predstavljaju minimalnu vrednost, prvi kvartil, medijanu, treći kvartil, maksimalnu vrednost, donju i gornju granicu 95% intervala poverenja za medijanu.

Poglavlje 7

Uslovi za završetak simulacije

U prethodnom poglavlju smo videli da statistika izlaza simulacije zavisi od broja replikacija; sada dajemo odgovor na pitanje koliko dugo po ulasku u ustaljeni režim treba izvršavati simulaciju da bismo dobili verodostojnu ocenu parametara.

Uslov za završetak simulacije se definiše pre njenog početka. Postoje dve kategorije ovih uslova, sistemski i statistički.

7.1 Sistemski uslovi

Najočigledniji uslovi za završetak simulacije su tzv. sistemski uslovi, koji su tako nazvani jer se zasnivaju na vrednostima parametara simulacionog okruženja.

Verovatno najčešće korišteni uslov podrazumeva zadavanje vremena tokom koga će se simulacija izvršavati; po isticanju ovoga vremena, simulacija se prekida. Sličan pristup podrazumeva da se simulacija prekida po dostizanju zadatog broja replikacija. Dobra strana ovih pristupa je to što unapred znamo koliko dugo će se simulacija izvršavati, ali nismo sigurni u to koliko će se slučajnih događaja generisati tokom njenog trajanja; kao što ćemo videti nešto kasnije, ovo za posledicu ima i nesigurnost u kvalitet tako dobijenih rezultata. Takođe, isticanje zadatog vremena ne mora koincidirati sa završetkom procesa obrade u sistemu (na primer, može se desiti da po isteku trajanja simulacije u radionici servisnog sistema ostanu korisnici čija je obrada započeta, a nije dovršena), pa se postavlja pitanje na koji način pri obradi rezultata uzeti u obzir nedovršene procese.

Simulacija se može prekinuti i kada se realizuje neki događaj (npr. kada iz čvora telekomunikacione mreže bude izašao hiljaditi paket), kada sistem uđe u zadato stanje (npr. kada se bude popunio neki bafer), ili kada statistički brojači dostignu zadate vrednosti (npr. kada ukupno vreme obrade paketa u čvoru dostigne vrednost od 10 minuta). Zajedničko za sve ove pristupe je da daju nešto bolji uvid u stanje simuliranog sistema,

ali ne znamo unapred koliko će vremena trebati da bi se oni ispunili.

7.2 Statistički uslovi

Statistički uslovi se zasnivaju na zadavanju kvaliteta rezultata. Simulacija se ovde prekida onda kada se postigne željena preciznost ocene parametara simuliranog sistema.

Neka se vrednost parametra X ocenjuje preko matematičkog očekivanja serije n ishoda,

$$\overline{X}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Podsetimo se izraza za interval poverenja matematičkog očekivanja u slučaju kada varijansa nije poznata (strana 75):

$$\left[\overline{X}(n) - t_{n-1,\,1-\alpha/2}\frac{s(n)}{\sqrt{n}},\ \overline{X} + t_{n-1,\,1-\alpha/2}\frac{s(n)}{\sqrt{n}}\right],$$

gde je $1-\alpha$ nivo poverenja i t kvantil Studentove t-raspodele sa $n-1$ stepenom slobode. Neka je, dalje, μ matematičko očekivanje slučajne promenljive X.

7.2.1 Apsolutna greška

Apsolutna greška izlaza simulacije definiše se kao

$$\beta = \left|\overline{X}(n) - \mu\right|. \tag{7.1}$$

Ako je konvergencija izlaza ka ustaljenom režimo monotona, tj. ako nema oscilatornih procesa, jasno je da će se apsolutna greška smanjivati s povećavanjem broja replikacija.

Neka se simulacija ponavlja sve dok poluširina intervala poverenja s nivoom $1-\alpha$ ne postane manja ili jednaka β:

$$\begin{aligned} 1-\alpha &\approx \mathrm{P}\left(\overline{X}(n) - \xi \leqslant \mu \leqslant \overline{X}(n) + \xi\right) = \\ &= \mathrm{P}\left(\left|\overline{X}(n) - \mu\right| \leqslant \xi\right) \leqslant \\ &\leqslant \mathrm{P}\left(\left|\overline{X}(n) - \mu\right| \leqslant \beta\right), \end{aligned} \tag{7.2}$$

gde je sa ξ označena poluširina intervala poverenja,

$$\xi = t_{n-1,\,1-\alpha/2}\frac{s(n)}{\sqrt{n}}. \tag{7.3}$$

Vidimo da se apsolutna greška nalazi unutar zadate granice s verovatnoćom $1-\alpha$.

Opisani postupak bismo u praksi sproveli na sledeći način: prvo se izvede test-serija od n replikacija, iz kojih se oceni varijansa. Potom se proceni potreban broj replikacija za dostizanje željene tečnosti kao

$$n^*(\beta) = \min\left\{ i \geqslant n \mid t_{i-1,\,1-\alpha/2}\sqrt{\frac{s^2(n)}{i}} \leqslant \beta \right\}. \tag{7.4}$$

Gornju nejednakost možemo rešavati iterativno, ili t-raspodelu možemo aproksimirati normalnom; u tome slučaju, dobićemo

$$i \geqslant s^2(n)\left(\frac{z_{1-\alpha/2}}{\beta}\right)^2, \tag{7.5}$$

gde je z odgovarajući kvantil standardne normalne raspodele.

7.2.2 Relativna greška

Relativna greška izlaza simulacije definiše se kao

$$\gamma = \left|\frac{\overline{X}(n) - \mu}{\mu}\right|. \tag{7.6}$$

Neka se simulacija ponavlja sve dok poluširina intervala poverenja za matematičko očekivanje, podeljena sa $|\mu|$, ne postane manja ili jednaka γ. Tada će važiti:

$$\begin{aligned}
1-\alpha \approx \mathrm{P}&\left(\left|\frac{\overline{X}(n)-\mu}{\overline{X}(n)}\right| \leqslant \frac{\xi}{|\overline{X}(n)|}\right) \leqslant \\
\leqslant \mathrm{P}&\left(\left|\frac{\overline{X}(n)-\mu}{\overline{X}(n)}\right| \leqslant \gamma\right) = \\
= \mathrm{P}&\left(\left|\overline{X}(n)-\mu\right| \leqslant \gamma\left|\overline{X}(n)\right|\right) = \\
= \mathrm{P}&\left(\left|\overline{X}(n)-\mu\right| \leqslant \gamma\left|\overline{X}(n)-\mu+\mu\right|\right) \leqslant \\
\leqslant \mathrm{P}&\left(\left|\overline{X}(n)-\mu\right| \leqslant \gamma\left|\overline{X}(n)-\mu\right| + \gamma|\mu|\right) = \\
= \mathrm{P}&\left(\left|\overline{X}(n)-\mu\right|(1-\gamma) \leqslant \gamma|\mu|\right) = \\
= \mathrm{P}&\left(\left|\frac{\overline{X}(n)-\mu}{\mu}\right| \leqslant \frac{\gamma}{1-\gamma}\right).
\end{aligned} \tag{7.7}$$

Vidimo da se ovako s verovatnoćom $1-\alpha$ postiže relativna greška ne veća od $\frac{\gamma}{1-\gamma}$.

U praksi bismo prvo izveli test-seriju od n replikacija, iz kojih bismo ocenili varijansu izlaza. Potreban broj replikacija za dostizanje željene tačnosti procenili bismo kao

$$n^*(\gamma) = \min\left\{ i \geqslant n \mid t_{i-1,\,1-\alpha/2}\frac{\sqrt{\frac{s^2(n)}{i}}}{\left|\overline{X}(n)\right|} \leqslant \frac{\gamma}{1-\gamma} \right\}. \tag{7.8}$$

Gornju nejednakost možemo rešavati iterativno, ili t-raspodelu možemo aproksimirati normalnom; u tom slučaju, dobićemo

$$i \geqslant s^2(n) \left(\frac{z_{1-\alpha/2}}{\frac{\gamma}{1-\gamma} \overline{X}(n)} \right)^2 . \tag{7.9}$$

Drugačiji pristup je tzv. sekvencijalni postupak, u kome se posle svake replikacije iznova ocenjuje varijansa. On je opisan sledećim koracima:

(0) Izvrši se $n_0 \geqslant 2$ replika simulacije,
(1) $n := n_0$,
(2) Izračunaju se $\overline{X}(n)$ i $\xi(n, \alpha)$,
(3) Ako je $\dfrac{\xi(n,\alpha)}{\left|\overline{X}(n)\right|} \leqslant \dfrac{\gamma}{1-\gamma}$, $\overline{X}(n)$ je tražena ocena; u suprotnom, $n := n+1$ i vraća se na korak (2).

7.3 Primer

Posmatrajmo ponovo integral 5.6. Cilj nam je da s nivoom poverenja od 95% ($\alpha = 0{,}05$) izračunamo ovaj integral s apsolutnom greškom ne većom od $\beta = 0{,}005$, ili relativnom greškom ne većom od $\gamma = 0{,}5\%$.

U programu GNU Octave smo izvršili test-seriju od $n = 50$ replikacija, na kojoj smo ocenili vrednosti $\overline{X}(n) = 0{,}35743$ i $s^2(n) = 0{,}21376$. Odavde smo dobili da se potrebna tačnost ostvaruje za barem 32848 replikacija ako je zadata apsolutna greška, ili za 254544 replikacije ako je zadata relativna greška. U prvom slučaju smo dobili rezultat $J = 0{,}33376$, čija je apsolutna greška 0,00043, dok je rezultat u drugom slučaju 0,33453, pa je relativna greška 0,36%.

Primenom sekvencijalnog metoda, za 221290 replikacija smo dobili $J = 0{,}33213$, pa je relativna greška iznosila 0,36%.

Poglavlje 8

Tehnike za smanjenje varijanse izlaza

Kada bismo pri izvršavanju simulacije mogli smanjiti varijansu izlaznog slučajnog procesa, tada bismo parametre performansi analiziranog sistema mogli oceniti sa zadatom tačnošću na izlaznom uzorku manjeg obima, tj. iz manjeg broja replikacija simulacije. Po cenu njene nešto složenije pripreme, ovako bismo mogli uštedeti vreme potrebno za izvođenje simulacije.

U nastavku ovoga poglavlja, opisaćemo nekoliko tehnika za smanjenje varijanse rezultata simulacije.

8.1 Antipodni brojevi

Ideja ove metode je da se izvode parovi replikacija simulacije, pri čemu se kao ulazni podaci u parovima koriste komplementarni slučajni brojevi. Na primer, ako se u jednoj replikaciji koristi broj U_k koji je dobijen iz uniformne raspodele na intervalu $[0,1]$, u drugoj će se *za istu svrhu* koristiti broj $1-U_k$, koji pripada istoj raspodeli.

Označimo slučajni izlazni proces koji se dobija u prvoj replikaciji para sa $X_1^{(1)}, X_2^{(1)}, \ldots, X_n^{(1)}$ a slučajni proces dobijen u drugoj replikaciji para sa $X_1^{(2)}, X_2^{(2)}, \ldots, X_n^{(2)}$. Očigledno, za svako $i,j = 1,2,\ldots,n$ važi $\mathrm{E}X_i^{(1)} = \mathrm{E}X_i^{(2)}$, pri čemu $X_i^{(1)}$ i $X_i^{(2)}$ nisu nezavisne veličine, ali parovi $(X_i^{(1)}, X_i^{(2)})$ i $(X_j^{(1)}, X_j^{(2)})$ za $i \neq j$ jesu.

Formirajmo sada usrednjen izlazni proces

$$X_i = \frac{X_i^{(1)} + X_i^{(2)}}{2}, \quad i = 1,2,\ldots,n. \tag{8.1}$$

Nepristrasna ocena matematičkog očekivanja ovakvog izlaza je

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{8.2}$$

Njena varijansa je

$$\begin{aligned}\operatorname{Var}\overline{X} &= \frac{1}{n^2}\operatorname{Var}\sum_{i=1}^{n} X_i = \frac{1}{n^2}\sum_{i=1}^{n}\operatorname{Var} X_i = \\ &= \frac{1}{n}\operatorname{Var} X_i = \frac{1}{n}\operatorname{Var}\frac{X_i^{(1)} + X_i^{(2)}}{2} = \\ &= \frac{\operatorname{Var} X_i^{(1)} + \operatorname{Var} X_i^{(2)} + 2\operatorname{Cov}(X_i^{(1)}, X_i^{(2)})}{4n}. \end{aligned} \tag{8.3}$$

Kada bi $X_i^{(1)}$ i $X_i^{(2)}$ bile nezavisne veličine, njihova kovarijansa bila bi jednaka nuli i imali bismo standardan rezultat za usrednjavanje $2n$ ishoda. Ako bismo, pak, uveli negativnu korelaciju dvaju izlaznih procesa, tada bi njihova kovarijansa bila negativna, što bi dovelo do željenog smanjenja varijanse usrednjenog izlaznog procesa.

Već iz ove jednostavne analize vidimo da je fundamentalan uslov koji treba ispuniti da bi antipodni brojevi vodili smanjenju varijanse taj da odziv sistema (tj. simulacionog modela) na neki ulaz bude monoton; u suprotnom, primenom antipodnih brojeva varijansa izlaza se može i povećati. Ako u modelu postoje ulazi s nemonotonim odzivom, tada će u nekim slučajevima biti bolje da se na njih ne primijeni ovaj metod, već npr. klasičan Monte Carlo.

Napomenimo da je bitno i da se ostvari sinhronizacija slučajnih brojeva korišćenih u parovima replikacija. Parovi komplementarnih (antipodnih) brojeva moraju se u parovima replikacija koristiti za identične svrhe – npr. u simulaciji ARQ procedura, u prvoj replikaciji se broj U_k koristi kao verovatnoća pogrešnog prijema k-tog okvira, dok se u drugoj replikaciji za to koristi broj $1 - U_k$.

Pitanje na koje nije jednostavno dati opšti odgovor je kako treba postupiti u slučaju kada simulacioni model koristi više slučajnih procesa kao ulaze. Idealno bi bilo kada bi se u parovima replikacija na svaki ulaz doveo komplementaran broj. U slučaju kada odziv sistema nije monoton na sve ulaze, bolje je da se antipodni brojevi primene samo na one ulaze koji su monotoni, a da se na nemonotone primeni klasična Monte Carlo simulacija.

8.2 Inverzni slučajni procesi

Negativna korelisanost između parova ishoda može se uvesti i na drugi način, korišćenjem veza između raspodela koje odgovaraju ulaznim veličinama. Ilustrovaćemo to na primeru servisnog sistema $M/M/1$.

Proces dolazaka korisnika u sistem $M/M/1$ je Poissonov, dok je vreme obrade eksponencijalno raspodeljeno. Između ovih dveju raspodela postoji fundamentalna veza, jer je vreme između dvaju sukcesivnih Poissonovih događaja eksponencijalno raspodeljeno.

Neka se na početku generišu dva niza eksponencijalno raspodeljenih slučajnih brojeva. U prvoj seriji replikacija, jedan od ovih nizova se koristi sa značenjem vremena međudolazaka korisnika, a drugi sa značenjem trajanja njihove obrade; u drugoj seriji replikacija, nizovi će obrnuti uloge. Rezultati (npr. za vreme zadržavanja korisnika u sistemu) će se agregirati u usrednjeni proces prema (8.1), na kome će se oceniti matematičko očekivanje. Na ovakav način, varijansa izlaza se može smanjiti za 65%.

8.3 Zajednički slučajni brojevi

Čest slučaj u praksi je da simulacijom treba utvrditi koja od alternativnih konfiguracija sistema ima bolje performanse. Posmatrajmo dve takve konfiguracije i označimo slučajni izlazni proces koji se dobija simulacijom prve sa $X_1^{(1)}, X_2^{(1)}, \ldots, X_n^{(1)}$ a slučajni proces dobijen simulacijom druge konfiguracije sa $X_1^{(2)}, X_2^{(2)}, \ldots, X_n^{(2)}$. Odluku o tome koja je konfiguracija bolja donosimo na osnovu ocene statistike

$$\theta = \mathrm{E}X^{(1)} - \mathrm{E}X^{(2)}. \tag{8.4}$$

Metodološki bi bilo najjednostavnije da izvršimo dve *nezavisne* serije simulacija, iz kojih bismo ocenili $\hat{\theta}_1 = \mathrm{E}X^{(1)}$ i $\hat{\theta}_2 = \mathrm{E}X^{(2)}$, pa potom formirali njihovu razliku $\hat{\theta} = \hat{\theta}_1 - \hat{\theta}_2$. Varijansa ovakve ocene bi bila

$$\operatorname{Var}\hat{\theta} = \operatorname{Var}\hat{\theta}_1 + \operatorname{Var}\hat{\theta}_2. \tag{8.5}$$

Ispitajmo da li je moguće smanjiti varijansu ocene θ. Formirajmo novi izlazni proces

$$Y_i = X_i^{(1)} - X_i^{(2)}, \quad i = 1, 2, \ldots, n. \tag{8.6}$$

Nepristrasna ocena njegovog matematičkog očekivanja je

$$\overline{Y} = \hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} Y_i, \tag{8.7}$$

dok je njena varijansa

$$\operatorname{Var}\overline{Y} = \operatorname{Var}\hat{\theta} = \frac{\operatorname{Var}X_i^{(1)} + \operatorname{Var}X_i^{(2)} - 2\operatorname{Cov}(X_i^{(1)}, X_i^{(2)})}{n}. \tag{8.8}$$

Za razliku od metode antipodnih slučajnih brojeva u kojoj smo uvodili negativnu korelaciju izlaznih procesa, ovde nam je cilj da pozitivno korelišemo procese $X_i^{(1)}$ i $X_i^{(2)}$. To možemo izvesti tako što ćemo na primer koristiti iste ulazne nizove slučajnih brojeva da bismo izvršili dve serije simulacija. Ideja koja leži u osnovi ovoga pristupa je da se sistemi uporede pod (približno) jednakim simulacionim uslovima.

8.4 Indirektne ocene

Metodu indirektnih ocena parametara ilustrovaćemo na primeru servisnog sistema $GI/G/m$.

Zadržavanje i-tog korisnika u sistemu jednako je zbiru vremena koje korisnik provede čekajući u čekaonici i vremena obrade u radionici,

$$t_i = q_i + s_i. \tag{8.9}$$

Direktne ocene parametara ovoga sistema su

$$\hat{T}_Q = \frac{1}{n}\sum_{i=1}^{n} q_i, \tag{8.10}$$

$$\hat{T} = \frac{1}{n}\sum_{i=1}^{n} t_i, \tag{8.11}$$

$$\hat{N}_Q = \frac{1}{t_{sim}}\int_0^{t_{sim}} N_Q(\tau)\,\mathrm{d}\tau, \tag{8.12}$$

$$\hat{N} = \frac{1}{t_{sim}}\int_0^{t_{sim}} N(\tau)\,\mathrm{d}\tau. \tag{8.13}$$

Pretpostavljeno je da simulacija traje od 0 do t_{sim}, tj. dok se ne bude obradilo n korisnika.

S druge strane, prosečno vreme zadržavanja korisnika u sistemu može se oceniti i kao

$$\tilde{T} = \hat{T}_Q + \mathrm{E}T_S, \tag{8.14}$$

pri čemu je očekivanje trajanja obrade, $\mathrm{E}T_S$, unapred poznato, jer trajanja obrade korisnika generišemo iz raspodele s poznatom srednjom vrednošću. Izraz (8.14) je primer indirektne aditivne ocene. Može se pokazati (ali to nije jednostavno) da ovakva indirektna ocena ima manju varijansu nego direktna, prema izrazu (8.11). Ovo je u neku ruku i očekivano, jer $\tilde{T}$ ocenjujemo samo iz jedne slučajne promenljive (q_i), a $\hat{T}$ iz dve (q_i i s_i).

Za ovaj primer postoji još nekoliko indirektnih estimatora. Prema Littleovoj teoremi je

$$N_Q = \lambda T_Q \tag{8.15}$$

i

$$N = \lambda T, \tag{8.16}$$

gde je λ protok dolazaka korisnika u sistem. Odavde se može izvesti multiplikativna indirektna ocena prosečnog broja korisnika u čekaonici,

$$\tilde{N}_Q = \lambda \hat{T}_Q, \tag{8.17}$$

koja je bolja od direktne ocene, date izrazom (8.12).

Još jedna multiplikativna indirektna ocena smanjene varijanse je

$$\tilde{N} = \lambda \tilde{T} = \lambda\left(\hat{T}_Q + \mathrm{E}T_S\right). \tag{8.18}$$

Korišćenjem ovih indirektnih ocena, za sistem $M/G/1$ se može postići smanjenje varijanse do 22%, naročito za male vrednosti iskorišćenosti servera.

8.5 Kontrolne promenljive

Neka želimo da ocenimo matematičko očekivanje izlaznog slučajnog procesa X, $\theta = \mathrm{E}X$. Neka je Y još jedan izlazni slučajni proces (ili neka funkcija X) čije je matematičko očekivanje poznato. Nije teško videti da statistika

$$Z_c = X + c(Y - \mathrm{E}Y), \tag{8.19}$$

u kojoj je $c \in \mathrm{R}$ predstavlja nepristrasnu ocenu θ. Varijansa ovakve ocene je

$$\operatorname{Var} \overline{Z}_c = \operatorname{Var} X + c^2 \operatorname{Var} Y + 2c \operatorname{Cov}(X, Y). \tag{8.20}$$

Gornji izraz dostiže minimalnu vrednost za

$$c^* = -\frac{\operatorname{Cov}(X, Y)}{\operatorname{Var} Y} \tag{8.21}$$

i ona iznosi

$$\begin{aligned} \operatorname{Var} \overline{Z}_{c^*} &= \operatorname{Var} X - \frac{(\operatorname{Cov}(X, Y))^2}{\operatorname{Var} Y} = \\ &= \operatorname{Var} \hat{\theta} - \frac{(\operatorname{Cov}(X, Y))^2}{\operatorname{Var} Y}. \end{aligned} \tag{8.22}$$

Vidimo da se varijansa smanjuje ako je $\operatorname{Cov}(X, Y) \neq 0$. Veličina Y uobičajeno se zove kontrolnom promenljivom za X.

U praktičnim primenama, prvo se izvrši test-serija od p replikacija iz kojih se oceni kovarijansa:

$$\widehat{\operatorname{Cov}}(X, Y) = \frac{\displaystyle\sum_{j=1}^{p} \left(X_j - \frac{1}{p} \sum_{i=1}^{p} X_i \right) (Y_j - \mathrm{E}Y)}{p - 1}. \tag{8.23}$$

U programima GNU Octave i Python, matrica kovarijanse slučajnih vektora se ocenjuje naredbom `cov`. Da bi se rezultat normalizovao sa $p-1$, a ne sa p, u Pythonu je potrebno navesti i vrednost opcionog argumenta `ddof = 1`.

Ukoliko nije poznata varijansa kontrolne promenljive, treba je oceniti iz rezultata test--serije. Tek posle toga se određuje optimalna vrednost parametra c, prema (8.21) i izvrši se glavna serija replikacija.

Gornji zaključci se mogu uopštiti tako što će se posmatrati više od jedne kontrolne promenljive, na primer njih m. U tome slučaju će biti

$$Z_c = X + \sum_{i=1}^{m} c_i (Y_i - \mathrm{E}Y_i), \tag{8.24}$$

pa je

$$\operatorname{Var} \overline{Z}_c = \operatorname{Var} X + 2 \sum_{i=1}^{m} c_i \operatorname{Cov}(X, Y_i) + \sum_{i=1}^{m} \sum_{j=1}^{m} \operatorname{Cov}(Y_i, Y_j). \tag{8.25}$$

Optimalne vrednosti koeficijenata c_i mogu se naći izjednačavanjem parcijalnih izvoda ovoga izraza s nulom; podesniji način je da se one procene kao $\hat{c}_i^* = -b_i$, gde su b_i rešenja linearne regresije

$$X = a + \sum_{i=1}^{m} b_i Y_i \tag{8.26}$$

u smislu najmanjeg srednjeg kvadratnog odstupanja.

8.6 Primer

Izračunajmo numeričkim putem vrednosti integrala

$$I = \int_0^{\pi/2} \sin(x)\,\mathrm{d}x, \quad J = \int_0^{\pi} \sin(x)\,\mathrm{d}x.$$

Oba integrala su analitički rešiva, pri čemu je $I = 1$ i $J = 2$. Kod prvog integrala, podintegralna funkcija je monotona na intervalu integracije, dok kod drugog nije.

Integrale ćemo numerički rešiti na dva načina, klasičnom Monte Carlo simulacijom u $2n$ tačaka i metodom antipodnih brojeva u dva puta po n tačaka. Dobijeni rezultati, za $n = 10$, dati su u tabeli.

Tabela 8.1: Rezultati izračunavanja.

	I		J	
	μ	s^2	μ	s^2
Monte Carlo	1,05366	0,22066	2,10803	0,81676
Antipodni brojevi	0,99189	0,00872	1,93370	1,10166

Vidimo da je, kao što smo i očekivali, metoda antipodnih brojeva dovela do značajnog smanjenja varijanse za integral I, dok je kod integrala J imala nešto lošije rezultate u odnosu na klasični Monte Carlo.

GNU Octave kod koji izračunava prvi integral dat je u nastavku.

```
N = 10;
X = unifrnd(0, pi/2, N, 1);
Q = sin(X)*pi/2;
W = sin(pi/2 - X)*pi/2;
Y = (Q + W)/2;
X = unifrnd(0, pi/2, 2*N, 1);
Q = sin(X)*pi/2;
[mean(Y), var(Y), mean(Q), var(Q)]
```

Poglavlje 9

Simulacija retkih događaja

U ovome poglavlju ćemo se upoznati s metodama za simuliranje događaja čija je verovatnoća realizacije toliko mala da direktna primena Monte Carlo simulacije ne bi bila efikasna.

9.1 Uvod

Iako ne postoji formalna definicija, smatra se da je neki događaj *redak* ako se realizuje s verovatnoćom koja ne premašuje vrednost 10^{-9}. Iako se ovakvi događaji retko javljaju, njihove posledice mogu biti dalekosežne; ilustrativni primeri su npr. kvarovi u nuklearnim elektranama ili na raketama.

Podsetimo se rezultata koje smo u odeljku 6.2.8 izveli za slučaj klasičnog Monte Carlo estimatora. Neka se verovatnoća realizacije događaja A, $y = \mathrm{P}(A)$ određuje Monte Carlo simulacijom. Nepristrasna ocena ove verovatnoće je

$$\overline{Y}_{MC} = \frac{1}{N} \sum_{i=1}^{N} e_i, \tag{9.1}$$

gde je e_i indikator događaja A,

$$e_i = \begin{cases} 1, & \text{ako je u } i\text{-toj replikaciji nastupio događaj } A \\ 0, & \text{u suprotnom} \end{cases}. \tag{9.2}$$

Varijansa ovoga estimatora je

$$\sigma^2_{MC} = \operatorname{Var} \overline{Y}_{MC} = \frac{y(1-y)}{N}, \tag{9.3}$$

dok je dvostrani interval poverenja s nivoom $1-\alpha$

$$\left(\overline{Y}_{MC} - z_{1-\alpha/2} \sqrt{\frac{y(1-y)}{N}},\ \overline{Y}_{MC} + z_{1-\alpha/2} \sqrt{\frac{y(1-y)}{N}} \right), \tag{9.4}$$

gde je z kvantil standardne normalne raspodele.

Ako se zahteva da maksimalna relativna poluširina intervala poverenja ne premaši vrednost β, odavde se može proceniti potreban broj replikacija Monte Carlo eksperimenta,

$$N \geqslant \frac{z_{1-\alpha/2}^2}{\beta^2} \frac{1-y}{y}. \tag{9.5}$$

Na primer, za nivo poverenja 99 % ($\alpha = 0{,}01$), $\beta = 0{,}1$ (10 %) i $y = 10^{-9}$, trebalo bi $N \geqslant 6{,}64 \cdot 10^{11}$ replikacija, što ne samo da bi dugo trajalo, već bi zahtevalo angažovanje značajnih memorijskih i procesorskih resursa. U nastavku ćemo razmotriti dve metode za efikasniju simulaciju retkih događaja.

9.2 Transformacija raspodele

Transformacija raspodele (engl. *importance sampling*) se zasniva na činjenici da su retki događaji retki zato što se nalaze na „repu" svoje raspodele; cilj je promeniti raspodelu tako da se poveća verovatnoća njihove realizacije u Monte Carlo eksperimentima.

Neka je izlaz simuliranog sistema dat sa $y = h(x)$, gde je X ulazna slučajna promenljiva s funkcijom gustine verovatnoće $f(x)$. Neka je $f^*(x)$ nova funkcija gustine verovatnoće koja zadovoljava uslov

$$h(x)f(x) > 0 \quad \Rightarrow \quad f^*(x) > 0. \tag{9.6}$$

Očekivanje izlaza je

$$\begin{aligned} \overline{Y} = \mathrm{E}\left(h(x)\right) &= \int h(x)f(x)\,\mathrm{d}x = \\ &= \int h(x)\frac{f(x)}{f^*(x)}f^*(x)\,\mathrm{d}x = \\ &= \mathrm{E}^*\left(h(x)\frac{f(x)}{f^*(x)}\right), \end{aligned} \tag{9.7}$$

gde je sa E^* označeno matematičko očekivanje kada se kao funkcija gustine verovatnoće koristi f^*. Ovde se, dakle, simulacija sprovodi za neku drugu raspodelu ulazne promenljive (ili više njih), za koju posmatrani događaj više nije redak, pa se potom iz dobijenog rezultata preračunava izlaz originalnog sistema na sledeći način: ako je izvedeno N Monte Carlo eksperimenata s transformisanom raspodelom ulaznog procesa, tražena ocena je

$$\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} h(x_i)\frac{f(x_i)}{f^*(x_i)}. \tag{9.8}$$

Na primer, ako se ocenjuje verovatnoća događaja, biće $h(x_i) = e_i$, pa je

$$\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} e_i\frac{f(x_i)}{f^*(x_i)}. \tag{9.9}$$

Zbog toga što se primenjuje transformisana raspodela, događaji od interesa će se javljati dovoljno često da bi se mogli prebrojati na manjoj seriji replikacija. Vraćanje na originalnu raspodelu vrši se prilikom ocene očekivanja izlaza, putem količnika

$$L(x) = \frac{f(x)}{f^*(x)}, \tag{9.10}$$

koji se naziva količnikom verodostojnosti.

Varijansa ovakvog estimatora je

$$\operatorname{Var} \overline{Y} = \frac{1}{N} \left(\mathrm{E}^* \left(h(x) L(x) \right)^2 - y^2 \right). \tag{9.11}$$

U posebnom slučaju kada je

$$f^*(x) = \frac{h(x) f(x)}{y}, \tag{9.12}$$

odnosno, ako se ocenjuje verovatnoća događaja

$$f^*(x) = \begin{cases} \frac{f(x)}{y}, & \text{ako } x \text{ označava povoljan ishod} \\ 0, & \text{u suprotnom} \end{cases}, \tag{9.13}$$

varijansa estimatora jednaka je nuli, pa bi se tačna ocena mogla dobiti iz samo jedne replikacije. Ovakav pristup nije primenljiv u praksi, jer zahteva apriorno poznavanje tačne vrednosti y, koja se upravo određuje simulacijom. Ono što se može ostvariti je dovoljno dobar suboptimalni rezultat, za koji raspodelu f^* treba odabrati tako da varijansa modifikovanog estimatora bude (značajno) manja od varijanse direktnog Monte Carlo estimatora, (9.3). Neke transformacije raspodele koje se često koriste u primenama su skaliranje,

$$f^*(x) = \frac{1}{\alpha} f\left(\frac{x}{\alpha}\right), \quad \alpha \in \mathbb{R}, \tag{9.14}$$

transliranje,

$$f^*(x) = f(x - T), \quad T \in \mathbb{R} \tag{9.15}$$

i eksponencijalna transformacija,

$$f^*(x) = \frac{e^{\vartheta x} f(x)}{M(\vartheta)}, \tag{9.16}$$

gde je sa $M(\vartheta)$ označena generišuća funkcija momenata raspodele f.

9.3 Razdvajanje događaja

Po metodi razdvajanja događaja (engl. *importance splitting*), retki događaj A, čija se verovatnoća realizacije određuje simulacijom, predstavlja se u vidu složenog događaja koji je sastavljen od događaja $A_{M+1} \subset A_M \subset \cdots \subset A_1$. Označimo sa $P_1 = \mathrm{P}(A_1)$ verovatnoću realizacije događaja A_1 i sa $P_i = \mathrm{P}(A_i | A_{i-1})$, $i = 2, \ldots, M+1$ uslovne

verovatnoće događaja u nizu. Cilj je izabrati putanju $A_1A_2 \dots A_{M+1}$ tako da pripadajući događaji ne budu retki, tj. da se njihove verovatnoće mogu efikasno oceniti Monte Carlo simulacijom.

Sama simuacija se realizuje na sledeći način. Na početku se generiše N Bernoullijevih slučajnih promenljivih, s verovatnoćom uspeha u pojedinačnom eksperimentu P_1. Ako se u ovoj seriji realizovao događaj A_1, generiše se novih R_1 Bernoullijevih slučajnih promenljivih s verovatnoćom uspeha P_2 i proveri da li se realizovao događaj A_2. Ponavljanjem postupka, dobija se

$$P = \mathrm{P}(A) = P_1 \cdots P_{M+1}, \tag{9.17}$$

pa je nepristrasna ocena P

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} \bar{P}_i = \frac{N_A}{NR_1 \cdots R_M}, \tag{9.18}$$

gde je N_A broj eksperimenata u kojima se realizovao događaj A. Matematičkom indukcijom se može pokazati da je varijansa ovoga estimatora

$$\operatorname{Var} \bar{P} = \frac{(P_1 \cdots P_{M+1})^2}{N} \sum_{i=0}^{M} \frac{1}{r_i} \left(\frac{1}{\mathrm{P}_{i+1|0}} - \frac{1}{\mathrm{P}_{i|0}} \right), \tag{9.19}$$

gde je

$$r_i = \begin{cases} R_1 \cdots R_i, & i = 1, \dots, M \\ 1, & i = 0 \end{cases} \tag{9.20}$$

i

$$\mathrm{P}_{i|0} = \begin{cases} P_1 \cdots P_i, & i = 1, \dots, M+1 \\ 1, & i = 0 \end{cases}. \tag{9.21}$$

„Cena" koju treba platiti da bi jedna realizacija slučajne promenljive prešla iz stanja A_{i-1} u stanje A_i je funkcija uslovne verovatnoće P_i, $h(P_i)$. Prosečna cena simulacije stoga je

$$C = N \sum_{i=0}^{M} r_i h(P_{i+1}) \mathrm{P}_{i|0}. \tag{9.22}$$

Za zadatu cenu, minimiziranjem varijanse estimatora se dobijaju sledeće optimalne vrednosti parametara simulacije:

$$P_i = \mathrm{P}^{1/(M+1)}, \quad i = 1, \dots, M+1, \tag{9.23}$$

$$R_i = \frac{1}{P_i}, \quad i = 1, \dots, M, \tag{9.24}$$

$$N = \frac{C}{(M+1)h\left(\mathrm{P}^{1/(M+1)}\right)}, \tag{9.25}$$

dok je

$$M = \left\lfloor \frac{\ln \mathrm{P}}{y_0} \right\rfloor \quad \vee \quad M = \left\lfloor \frac{\ln \mathrm{P}}{y_0} \right\rfloor - 1, \tag{9.26}$$

pri čemu je y_0 rešenje jednačine

$$(2(1-e^y)+y)\,h(e^y) - y(1-e^y)e^y h'(e^y) = 0. \tag{9.27}$$

U posebnom slučaju kada je $h(x) \equiv 1$, optimalne vrednosti parametara su $M = \lfloor -0{,}6275 \ln \mathrm{P} \rfloor - 1$ ili $M = \lfloor -0{,}6275 \ln \mathrm{P} \rfloor$, $R_i \approx 5$ i $P_i \approx \frac{1}{5}$.

9.4 Primer

Odredimo verovatnoću da se u stotinu bacanja fer novčića dobije tačno 80 „glava".

Problem se može rešiti analitički. Broj „glava" ima binomnu raspodelu, pa je verovatnoća da će se u 100 bacanja dobiti 80 „glava"

$$\mathrm{P} = \binom{100}{80} \left(\frac{1}{2}\right)^{80} \left(\frac{1}{2}\right)^{20} = 4{,}23 \cdot 10^{-10}. \tag{9.28}$$

Da bi se primenom klasične Monte Carlo simulacije registrovao jedan pozitivan ishod (80 „glava"), stotinu novčića bi u proseku trebalo baciti $N = 1/\mathrm{P} = 2{,}37 \cdot 10^9$ puta.

Primenimo transformaciju verovatnoće, što je ekvivalentno bacanju „nefer" novčića za koji je verovatnoća „glave" npr. $p = 0{,}8$. U programu GNU Octave ćemo izvršiti 9000 serija Monte Carlo eksperimenata i prebrojati pozitivne ishode:

```
p = 0.8;
y = binornd (100, p, 45, 200);
k = (y == 80);
out = sum(sum(k));
```

Verovatnoća pozitivnog ishoda s modifikovanom raspodelom ulaza je *out*/9000; da bismo dobili originalnu verovatnoću, treba da obračunamo i količnik verodostojnosti koji iznosi

$$L = \frac{\left(\frac{1}{2}\right)^{100}}{p^{80}(1-p)^{20}}. \tag{9.29}$$

U programu GNU Octave stoga ćemo kucati:

```
L = 0.5^100 / (p^80 * (1-p)^20);
P = out/(45*200) * L
```

Dobili smo rezultat $\mathrm{P} = 4{,}22 \cdot 10^{-10}$, koji se veoma malo razlikuje od tačnog.

Poglavlje 10

Simulaciona optimizacija

U ovome poglavlju ćemo se upoznati s primenom simulacije za optimizaciju sistema.

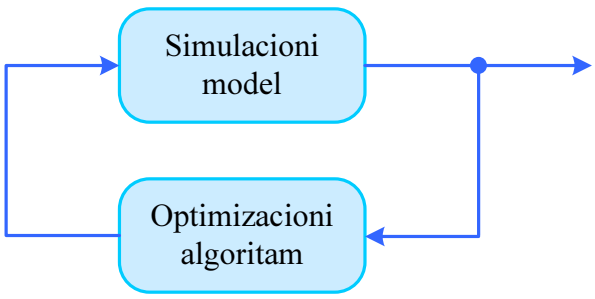
10.1 Uvod

Neka je data tzv. funkcija cilja $f : X \to \mathbb{R}$, gde je $X = (x_1 \dots x_m)$ vektor ulaznih promenljivih. Zadatak klasične optimizacije je da nađe X za koje funkcija cilja dostiže ekstremnu (npr. minimalnu) vrednost:

$$X_{opt} = \arg\min_{X \in \mathbb{X}} f(X), \tag{10.1}$$

pod ograničenjima $\mathbb{X} \subset \mathbb{R}^m$.

U literaturi je opisan veći broj optimizacionih metoda. Problem nastaje onda kada funkcija cilja uključuje i slučajne procese, kada je njen analitički oblik toliko složen da je nepodesan za praktične manipulacije, ili kada nije poznat u zatvorenoj formi; tada pribegavamo simulacionoj optimizaciji da bismo ocenili funkciju cilja i našli optimalnu dozvoljenu kombinaciju ulaznih parametara. Ovo je ilustrovano na slici 10.1.



Slika 10.1: Simulaciona optimizacija.

U opštem slučaju, funkcija cilja u simulacionoj optimizaciji je oblika

$$f(X) = \mathrm{E}\, g(X), \tag{10.2}$$

dok ograničenja mogu biti deterministička

$$u(X) < 0 \tag{10.3}$$

i stohastička:

$$v(X) = \mathrm{E}\, w(X) < 0, \tag{10.4}$$

ili

$$\mathrm{P}\left(v(X) < 0\right) > 1 - \alpha. \tag{10.5}$$

U daljem razmatranju ćemo razlikovati slučajeve kontinualnih i diskretnih ulaznih promenljivih.

10.2 Kontinualne ulazne promenljive

Ako su ulazne promenljive kontinualne, skup njihovih dozvoljenih vrednosti $\mathbb{X}$ biće beskonačan i neprebrojiv. U ovome slučaju se može primeniti iterativna metoda stohastičke aproksimacije, koja je analogna metodi najstrmijeg pada iz nelinearne optimizacije. Ako je $X^{(k)}$ vrednost ulaznog vektora u k-tom koraku algoritma, njegova nova vrednost biće

$$X^{(k+1)} = \prod_{\mathbb{X}} \left(X^{(k)} - \alpha_k \hat{\nabla} f(X^{(k)}) \right), \tag{10.6}$$

gde je sa $\hat{\nabla} f$ označena ocena gradijenta funkcije cilja, α_k je veličina koraka, dok je $\Pi_{\mathbb{X}}$ projekcija na dozvoljenu oblast $\mathbb{X}$. Da bi metoda asimptotski konvergirala ka stvarnom minimumu, potrebno je da koeficijenti α_k ispunjavaju uslove

$$\lim_{k \to \infty} \alpha_k = 0 \tag{10.7}$$

i

$$\sum_k \alpha_k < \infty. \tag{10.8}$$

Gradijent funkcije cilja može se numerički oceniti nekom od metoda iz poglavlja 4. Da bi se smanjio broj potrebnih izračunavanja i njima pridruženih simulacija, predložena je tzv. metoda simultanih perturbacija,

$$\left(\hat{\nabla} f(X) \right)_i = \frac{f(X + \Delta) - f(X - \Delta)}{2\delta_i}, \tag{10.9}$$

pri čemu je $\Delta = (\delta_1 \ldots \delta_m)$ vektor nezavisnih identično raspodeljenih slučajnih perturbacija, koje se u praksi najčešće biraju kao Bernoullijeve slučajne promenljive ($\delta_i \in \{\pm c_i\}$, s verovatnoćama 0,5).

10.3 Diskretne ulazne promenljive

Kada su ulazne promenljive diskretne, mogu se razlikovati dva podslučaja, u zavisnosti od toga da li je kardinalni broj skupa $\mathbb{X}$ mali ili veliki.

10.3.1 Slučaj malog kombinatornog prostora

Ako je broj dozvoljenih vrednosti ulaznog vektora X relativno mali, tada se mogu odrediti i uporediti vrednosti funkcije cilja za svaku od njih. Zbog postojanja stohastičkih procesa i činjenice da se radi o ocenama vrednosti, najbolje rešenje se neće odrediti prostim poređenjem, već statističkim testovima; ovo je tzv. metoda rangiranja i selekcije. Na primer, ako je tekuće „najbolje" rešenje X^*, neko drugo rešenje X se usvaja kao bolje ako je

$$\mathrm{P}\left(|f(X) - f(X^*)| < \delta\right) \geqslant 1 - \alpha, \tag{10.10}$$

gde je δ poluprečnik zone tolerancije i $1 - \alpha$ usvojeni nivo poverenja.

Da bi se omogućilo fer poređenje, pogodno je primeniti metodu zajedničkih slučajnih brojeva s kojom smo se upoznali na strani 95.

10.3.2 Slučaj velikog kombinatornog prostora

Kada zbog veličine kombinatornog prostora dozvoljenih rešenja nije izvodljivo ispitati svako od njih, mogu se primeniti metode determinističke optimizacije.

Slučajno pretraživanje

Ova metoda je opisana sledećim koracima:

0. Izabere se početno rešenje $X^{(0)}$, izvrši simulacija i ocene performanse sistema. Vrednost brojača k postavi se na nulu.
1. Iz susedstva tekućeg rešenja se izabere kandidat X^c, izvrši simulacija i ocene performanse sistema.
2. Ako kandidat zadovoljava usvojeni kriterijum prihvatanja, koji se zasniva na oceni performanse, tekuće rešenje se odbacuje, a prihvata se kandidat; u suprotnom, tekuće rešenje se ne odbacuje.
3. Ako je zadovoljen kriterijum za zaustavljanje, pretraživanje se prekida. U suprotnom, inkrementira se vrednost brojača k i algoritam se vraća na korak 1.

Simulaciono kaljenje

Simulaciono kaljenje je varijacija metode slučajnog pretraživanja; naziv je dobilo po analogiji s kaljenjem u metalurgiji, gde se zagrejane legure postepeno hlade, tako da dođu u stanje minimalne energije. Cilj simulacionog kaljenja je da algoritam konvergira ka globalnom optimumu, a ne da se pri pretraživanju kombinatornog prostora mogućih rešenja „zaglavi" u nekom od lokalnih. Osnovna razlika u odnosu na izvorni algoritam slučajnog pretraživanja je u koraku 2, jer se sada dopušta da se kandidatsko rešenje prihvati i u slučajevima kada je gore od tekućeg. U primenama se često koristi sledeći izraz za verovatnoću prihvatanja kandidatskog rešenja:

$$\mathrm{P} = \begin{cases} 1, & f(X^c) < f(X^{(k)}) \\ e^{\frac{f(X^c)-f(X^{(k)})}{T_k}}, & \text{u suprotnom} \end{cases}. \tag{10.11}$$

U skladu s objašnjenjem o nastanku naziva ove metode, parametar T_k se naziva temperaturom.

10.4 Primer

U ovome primeru ćemo pokazati da se u praktičnim primenama često do rešenja lakše može doći pomoću heuristika, nego direktnom primenom prethodno opisanih metoda.

Posmatrajmo optoelektronski prijemnik u sistemu za prenos digitalnog signala postupkom intenzitetske modulacije s direktnom detekcijom (IM-DD). Neka je verovatnoća slanja bita „0" p. Neka su, dalje, naponski nivoi bita „0" i „1" redom 0 i $U > 0$. Koristan signal i šum sada nisu nezavisni slučajni procesi: kada se šalje bit „0", standardna devijacija Gaussovog šuma je σ_0, a kada se šalje bit „1", $\sigma_1 > \sigma_0$. Potrebno je da odredimo optimalnu vrednost praga odlučivanja u prijemniku, $U_t \in [0, U]$, tako da verovatnoća donošenja pogrešne odluke bude minimalna.

Problem možemo rešiti na nekoliko načina.

Po prvom načinu, polazimo od izraza za verovatnoću greške,

$$\mathrm{P}_e = \frac{p}{\sqrt{2\pi}\sigma_0} \int_{U_t}^{+\infty} e^{-\frac{x^2}{2\sigma_0^2}} \,\mathrm{d}x + \frac{1-p}{\sqrt{2\pi}\sigma_1} \int_{-\infty}^{U_t} e^{-\frac{(x-U)^2}{2\sigma_1^2}} \,\mathrm{d}x, \tag{10.12}$$

nađemo njegov izvod po U_t i izjednačimo ga s nulom. Pre toga, podsetićemo se formule za diferenciranje po granici integrala:

$$\frac{\mathrm{d}}{\mathrm{d}x} \int_a^x f(t) \,\mathrm{d}t = f(x). \tag{10.13}$$

Sada je

$$\frac{\mathrm{d}P_e}{\mathrm{d}U_t} = -\frac{p}{\sqrt{2\pi}\sigma_0} e^{-\frac{U_t^2}{2\sigma_0^2}} + \frac{1-p}{\sqrt{2\pi}\sigma_1} e^{-\frac{(U_t-U)^2}{2\sigma_1^2}}. \tag{10.14}$$

Izjednačavanjem s nulom dobijamo

$$\frac{1-p}{\sigma_1} e^{-\frac{(U_t-U)^2}{2\sigma_1^2}} - \frac{p}{\sigma_0} e^{-\frac{U_t^2}{2\sigma_0^2}} = 0. \tag{10.15}$$

Ovu jednačino ćemo rešiti numerički.

Na drugi način, pošli bismo od izraza za verovatnoću greške, ovaj put napisanog kao

$$P_e = \frac{p}{2} \operatorname{erfc} \frac{U_t}{\sqrt{2}\sigma_0} + \frac{1-p}{2} \operatorname{erfc} \frac{U - U_t}{\sqrt{2}\sigma_1}, \tag{10.16}$$

tabelirali ga i očitali vrednost napona praga za koju verovatnoća greške dostiže minimum. Ako bismo želeli da povećamo preciznost rezultata, potom bismo funkciju (10.16) u okolini tečke „kandidata" za minimum tabelirali s manjim korakom U_t i ponovili pretraživanje.

Na kraju, ako bismo želeli da ostanemo verni simulacionom pristupu, slično kao u odeljku 5.1 bismo generisali sekvencu bita, dodali joj sekvencu šuma i ocenili verovatnoću greške za različite vrednosti napona praga odlučivanja. Poređenjem ocena verovatnoće greške, pronašli bismo napon praga za koji je ona minimalna. Da bi poređenje bilo fer, sekvence bita i šuma generisali bismo samo jednom, na početku, dok bismo performanse sistema (u ovom slučaju verovatnoću greške) ocenjivali za niz vrednosti U_t.

Ilustrujmo ova razmatranja na numeričkom primeru. Neka je $p = \frac{1}{2}$, $U = 1$ V, $\sigma_0 = 0{,}1$ V i $\sigma_1 = 0{,}25$ V. Problem rešavamo u programu GNU Octave.

U prvom slučaju, za numeričko rešavanje jednačine $f(x) = 0$ koristimo naredbu

```
[x, fval, info] = fsolve (fcn, x0)
```

Njeni ulazni argumenti su, redom, naziv funkcije i inicijalna tačka; izlazni argumenti su, redom, optimalna vrednost x, vrednost funkcije u ovoj tački i poruka o razlogu završetka izračunavanja.

Prvo ćemo definisati funkciju f:

```
function y = f(x)
y = .5/.25 * exp(-((1-x)/(sqrt(2)*.25))^2) - ...
 .5/.1 * exp(-(x/(sqrt(2)*.1))^2);
endfunction
```

zatim kucamo

```
[x, fval, info] = fsolve (@f, .5)
```

i kao rezultat dobijamo

```
x =  0.30810
fval = 7.6328e-017
info =  1
```

Poruka broj 1 znači da je rešenje konvergiralo, pa je optimalna vrednost napona praga 0,30810 V.

Na drugi način, kucali bismo

```
x = linspace(0, 1, 1000)';
pe = 0.25.*erfc(x/(0.1*sqrt(2))) + ...
 0.25.*erfc((1-x)/(0.25*sqrt(2)));
[q, w] = min(pe)
```

Kao odgovor, dobijamo

```
q =  0.0019276
w =  309
```

Konačno, kucamo `x(w)` i dobijamo da je optimalna vrednost napona praga 0,30831 V.

Simulacijom, problem rešavamo na sledeći način:

```
N = 1e5;
U = 1;
s0 = 0.1;
s1 = 0.25;
X = randint(N, 1, [0, 1]);
n0 = s0*randn(N, 1);
n1 = s1*randn(N, 1);
Y = X + X.*n1 + (1-X).*n0;
Ut = linspace(0, 1, 1000)';
for i = 1:size(Ut)(1)
Z = Y>=Ut(i);
Nerr = sum(X!=Z);
Pe(i) = Nerr/N;
endfor
```

Upitom `[q, w] = min(Pe)`, dobijamo odgovor

```
q =  0.0017400
w =  310
```

Da bismo očitali optimalnu vrednost napona praga, kucaćemo `x(w)`, pa ćemo dobiti da ona iznosi 0,30931 V.

Zanimljivo je primetiti da (10.15) ima analitičko rešenje (štaviše, ne samo jedno, već dva):

$$U_{t,1} = \frac{\sigma_0 \left(\sigma_1 \sqrt{U^2 - 2\left(\sigma_1^2 - \sigma_0^2\right) \ln \frac{(1-p)\sigma_0}{p\sigma_1}} - \sigma_0 U \right)}{\sigma_1^2 - \sigma_0^2}, \tag{10.17}$$

$$U_{t,2} = -\frac{\sigma_0 \left(\sigma_1 \sqrt{U^2 - 2\left(\sigma_1^2 - \sigma_0^2\right) \ln \frac{(1-p)\sigma_0}{p\sigma_1}} + \sigma_0 U \right)}{\sigma_1^2 - \sigma_0^2}. \tag{10.18}$$

Do ovih rezultata možemo doći npr. pomoću programa za simbolička izračunavanja Maxima. Fizičkog smisla ima rešenje (10.17), odakle uvrštavanjem zadatih brojčanih vrednosti dobijamo $U_t = 0{,}3080955$ V.

Čitaocima se prepušta da zaključe koji je način rešavanja ovoga problema najbolji, kako u pogledu tačnosti, tako i u pogledu ekonomičnosti.

Poglavlje 11

Simulacija telekomunikacionih mreža

Zbog njenog izuzetnog značaja i nekih metodoloških specifičnosti, ovo poglavlje u potpunosti posvećujemo razmatranju simulacije telekomunikacionih i računarskih mreža.

11.1 Uvod

Simulacija telekomunikacionih i računarskih mreža je primena simulacije s kojom se inženjeri telekomunikacija najčešće sreću u svom profesionalnom radu.

Razlozi za simulaciju mreža jednaki su opštim razlozima zbog kojih neki sistem analiziramo simulacijom, a koje smo diskutovali na početku ove knjige. Stvarne mreže često nisu dostupne za eksperimente (npr. zbog troškova opreme, ili zbog toga što ne smemo rizikovati da izazovemo poremećaj koji bi njihove korisnike ostavio bez servisa), pa stoga simulacijom analiziramo njihove surogate. Simulacijom možemo replicirati postojeće mreže i tako npr. ispitivati alternativne konfiguracije u cilju optimizacije ili provere otpornosti na neke vanredne događaje, kao što su ispad linka ili hakerski napad. Neke od ovih događaja koji su se desili u prošlosti možemo replicirati, da bismo videli – korak po korak – koje su posledice imali. Na kraju, simulacijom stičemo značajna saznanja u procesu razvoja novih mrežnih tehnologija, što se podjednako odnosi i na hardver i na protokole.

Najčešći cilj simulacije telekomuniakcionih mreža je određivanje kašnjenja, jittera, iskorišćenosti linka, efikasnosti neke protokolske procedure, verovatnoće sudara, verovatnoće retransmisije, verovatnoće odbacivanja paketa i sl. Po pravilu se radi o simulaciji diskretnih događaja koji su vezani za generisanje i prenos paketa. Da bi se ubrzala simulacija velikih mreža, primenjuju se i različite varijante paralelne i/ili distribuirane simulacije.

U opštem slučaju, telekomunikacione mreže se mogu simulirati u programskim jezicima opšte namene, kao što su npr. C, C++ i Python, matematičkim programima, kao što je

GNU Octave, ili u specijalizovanim mrežnim simulatorima. Poznati mrežni simulatori koji spadaju u kategoriju slobodnog softvera su *ns*-2, *ns*-3, GTNetS (Georgia Tech Network Simulator), SSFNet (Scalable Simulation Framework Network Models) i drugi. Izbor programa zavisi od karakteristika razmatrane mreže i problema koji se rešava.

11.2 Komponente simulacionog modela

Simulacioni model telekomunikacionih mreža u najopštijem slučaju obuhvata čvorove, interfejse, linkove, redove čekanja, protokole, pakete, aplikacije i pomoćne module. U nastavku ovoga odeljka ćemo razmotriti karakteristike svake od ovih komponenti ponaosob.

Čvorovi su svi elementi mreže koji šalju ili primaju pakete; to su, na primer korisnički računari, web serveri, habovi, ruteri, komutatori itd. Za opis objekata čvorova u simulacionom modelu su potrebni:

- grafovi protokola, koji predstavljaju logičke reprezentacije slojeva s opisom procedura za postupanje s paketima na tome sloju;
- mapa portova, tj. lista koja sadrži podatke o aplikacijama koje se izvršavaju u posmatranom krajnjem čvoru, s brojevima portova koji su im pridruženi. Mapa portova omogućava demultipleksiranje primljenih paketa ka odgovarajućim aplikacijama ili procesima;
- podaci o korišćenju resursa, kao što su npr. stanje baterije, iskorišćenost procesora, količina slobodnog memorijskog prostora i sl;
- fizička lokacija čvora, što je naročito važan podatak u bežičnim *ad hoc* mrežama i
- lista mrežnih interfejsa na posmatranom čvoru.

Mrežni interfejsi vode računa o razgraničenju paketa, pristupu sredini za prenos, detekciji greške i električnim signalima na fizičkom sloju. Atributi kojima su opisani objekti interfejsa su:

- indikator zauzetosti sredine za prenos;
- bafer paketa, u koji se smeštaju novopridošli paketi kada je sredina za prenos zauzeta i
- komunikacioni link, koji se pridružuje svakom mrežnom interfejsu.

U praktičnim implementacijama, indikator zauzetosti se najčešće realizuje u vidu „busy" flega.

Objekti linkova specificiraju fizičku povezanost elemenata mreže. Da bi se opisali, potrebni su:

- lista suseda, koja pokazuje koji su čvorovi direktno povezani sa čvorom koji je pridružen posmatranom linku;
- podatak o propagacionom kašnjenju na linku i
- podatak o kapacitetu linka.

Ovde ćemo dati dve napomene. Primetimo da je lista suseda u fiksnim mrežama skoro statična (menja se kada se neki uređaj uključi, isključi ili poveže na mrežu preko druge pristupne tačke), dok u mobilnim nije. Dalje, iako je u stvarnosti maksimalan protok na linku često određen interfejsom (npr. Ethernet kartica se programski konfiguriše za jedan od protoka 10 Mb/s ili 100 Mb/s), kapacitet linka se u simulaciji po pravilu pridružuje objektu linka.

Objekti redova čekanja se odnose na bafere. Za njihov opis treba poznavati:

- disciplinu opsluživanja paketa;
- kapacitet reda, u bajtovima ili paketima i
- listu paketa koji čekaju u redu.

Objekti protokola opisuju mesto protokola u slojevitom modelu (ISO-OSI, TCP-IP, ili hibridnom). Takođe opisuju kome protokolski entitet pruža servis i čiji servis koristi, kao i koje radnje u tome cilju preduzima. Konkretni detalji opisa objekata protokola i relevantne promenljive stanja zavise od karakteristika razmatranog protokola.

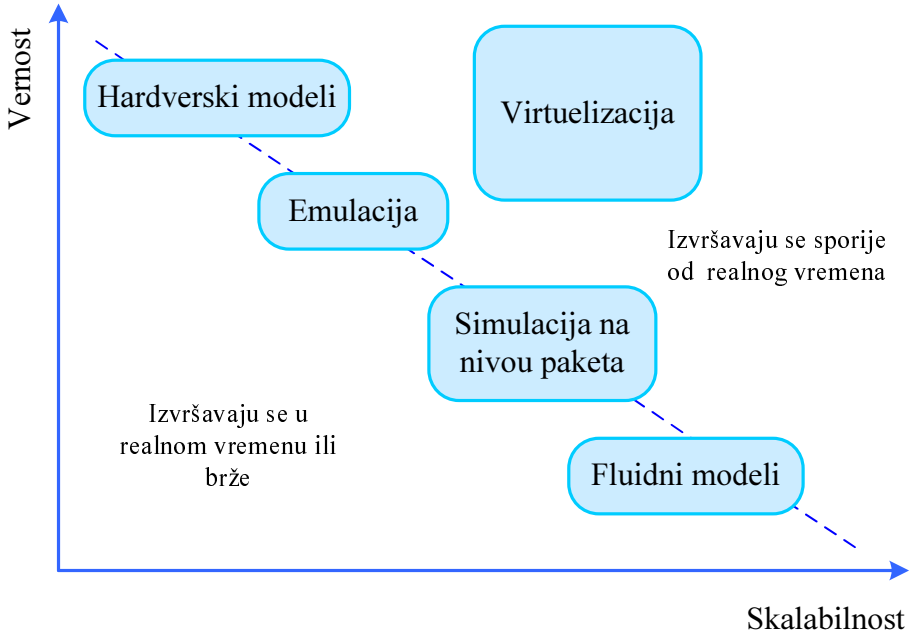
Podaci koje generišu krajnji sistemi se dele u pakete, koji se potom prenose kroz mrežu. Paketi se sastoje iz korisnog sadržaja i informacije koju mu dodaju protokoli; ova informacija se naziva „overhead" i po pravilu se nalazi u zaglavlju, a ponegde i u začelju paketa. Uobičajen pristup u simulacijama je da se posmatra samo „overhead", dok se koristan sadržaj paketa apstrahuje; to znači da je po pitanju korisnog sadržaja paketa samo bitno koliko ga ima (tj. kolika je veličina paketa), a ne šta on konkretno znači.

Objekti aplikacija se odnose na aplikacije koje se izvršavaju na krajnjim sistemima i koje predstavljaju pošiljaoce i primaoce paketa. I ovde se primenjuje izvesna apstrakcija, jer je sa stanovišta simulacije po pravilu samo bitno kakve su karakteristike prenošenog saobraćaja. Statističke karakteristike telekomunikacionog saobraćaja zavise od tipa aplikacije koja ga generiše (npr. da li je u pitanju klasična telefonija, FTP, web, VoIP ili Telnet). Simulacioni modeli aplikacija kao generatora telekomunikacionog saobraćaja mogu se napraviti na nekoliko načina: moguće je statistički opisati saobraćaj i potom generisati slučajne procese koji će ga emulirati; mogu se koristiti saobraćajne sekvence snimljene u stvarnoj mreži, koje će se reprodukovati u simulaciji, ili se aplikacioni softver može inkorporirati u simulacioni; ovaj potonji pristup se sve više koristi u razvoju i testiranju mrežnih protokola.

Pomoćni objekti prikupljaju podatke tokom izvršavanja simulacije. Po njenom završetku, obrađuju ove podatke i generišu izveštaj.

11.3 Simulacija fiksnih mreža

Bitni parametri simulacije procesa u fiksnim mrežama su skalabilnost izvršavanja, iskazana kroz maksimalan broj čvorova koji se mogu simulirati i vernost modela, iskazana količinom detalja koji se uzimaju u obzir. Različiti pristupi analizi fiksnih mreža ilustrovani su na slici 11.1 i razmotreni u nastavku ovoga odeljka.



Slika 11.1: Metode za analizu fiksnih mreža.

11.3.1 Hardverski modeli

Hardverski modeli najčešće predstavljaju funkcionalno umanjene verzije stvarnih uređaja, a ređe kompletnih mreža. Na tržištu se javljaju u vidu tzv. demonstracionih ploča i okruženja, kao i maketa za obrazovne svrhe. Odlikuju se velikom vernošću u modeliranju pojave i velikom brzinom rada, koja može premašiti brzinu njihovih „velikih" originala. Mane su im mala skalabilnost, jer nije moguće proširivanje funkcionalnosti dodavanjem novih softverskih ili hardverskih modula i, po pravilu, visoka cena. Upotreba hardverskih modela u simulaciji telekomunikacionih mreža stoga je ograničena na obrazovne ustanove.

11.3.2 Emulacija

Emulacija podrazumeva povezivanje simulirane mreže sa stvarnom. Emulacija se može realizovati prema dvama komplementarnim scenarijima. Po prvom, stvarni hardverski sistem je dominantan element modela; on generiše pakete koji se u realnom vremenu uvoze u simulaciono okruženje i potom obrađuju. Ovakav pristup se koristi pri

npr. verifikaciji protokola. Po drugom scenariju, koji se naziva i *hardware in the loop*, dominantan element modela je simulaciono okruženje. Obradu paketa sada preuzima hardverski segment, koji se često realizuje korišćenjem tzv. razvojnih ploča sa FPGA integrisanim kolima i koji zamenjuje neki od elemenata mreže iz simulacionog modela. Na ovaj način se proverava rad mrežnih uređaja tokom njihovog projektovanja i razvoja.

Bez obzira na to koji je scenario realizacije primenjen, da bi emulacija ispravno funkcionisala neophodno je uskladiti brzine rada stvarnog i simuliranog sistema.

11.3.3 Simulacija na nivou paketa

Simulacija na nivou paketa je pristup koji se najčešće koristi. Zasniva se na posmatranju razmene paketa između čvorova mreže koji su spojeni linkovima. Ovakvi modeli se lako razvijaju i dobro oponašaju rad realnih mreža, pa su dobijeni rezultati pravolinijski primenljivi u stvarnom svetu. Radi postizanja što veće vernosti, u nekim namenskim simulatorima se ide tako daleko da se koristi inkorporirani programski kod protokola. Najpoznatiji primeri simulatora mreža iz kategorije slobodnog softvera koji primenjuju simulaciju na nivou paketa su *ns*-2 i *ns*-3.

Najveći nedostatak simulacije na nivou paketa je velika kompleksnost izračunavanja za velike mreže. Problem je u efikasnom ažuriranju liste budućih događaja, pa se tako lako može desiti da se brže obradi paket, nego ova lista. Da bi se to prevazišlo, predloženi su računski efikasni algoritmi za umetanje događaja na listu i brisanje iz nje.

Još jedan problem simulacije na nivou paketa je održavanje tabela rutiranja u čvorovima mreže, za šta je potrebna velika memorija. Da bi se uštedelo na memorijskom prostoru, predložene su različite heuristike. Negde se primenjuje dinamički pristup, s inkrementalnim određivanjem ruta; takođe, često korišćene rute se mogu čuvati u keš-memoriji, dok se rute koje se ne koriste u simulaciji ni ne tabeliraju.

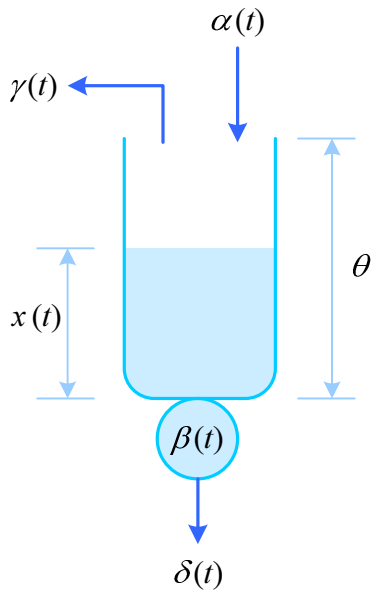
11.3.4 Fluidni modeli

Fluidni modeli ne posmatraju pojedinačne pakete, već agregirane tokove saobraćaja na linkovima. Fluidni model bafera prikazan je na slici 11.2.

Prema oznakama sa slike, stanje bafera opisano je jednačinama

$$\delta(t) = \begin{cases} \beta(t), & x(t) \neq 0 \\ \alpha(t), & \text{inače} \end{cases}, \tag{11.1}$$

$$\gamma(t) = \begin{cases} 0, & x(t) < \theta \\ \alpha(t) - \beta(t), & x(t) = \theta \end{cases}, \tag{11.2}$$



$\alpha(t)$ – ulazni protok
$\delta(t)$ – izlazni protok
$\beta(t)$ – maksimalan izlazni protok
θ – kapacitet bafera
$x(t)$ – popunjenost bafera
$\gamma(t)$ – protok gubitka

Slika 11.2: Fluidni model bafera.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \begin{cases} 0, & x(t) = 0,\ \alpha(t) < \beta(t) \\ 0, & x(t) = \theta,\ \alpha(t) \geqslant \beta(t)\ . \\ \alpha(t) - \beta(t), & \text{inače} \end{cases} \tag{11.3}$$

Nije teško videti da ovaj model odgovara *leaky bucket* uobličavaču saobraćaja.

Osnovni nedestatak fluidnog modela je u tome što ignoriše kašnjenje paketa.

11.3.5 Virtuelizacija

Virtuelizacija je novi pristup simulaciji mreža. Podrazumeva „podizanje" virtuelnog uređaja (ili čitave mreže) na računaru opšte namene. Virtuelizovani sistemi mogu ostvariti izuzetnu vernost, čak bolju nego kod hardverskih modela, uz veliku skalabilnost; takođe je moguće povezati virtuelni uređaj sa stvarnim, pa tako ostvariti jedan vid emulacije.

Mana virtuelizacije je to što ovakvi sistemi rade sporije od realnog vremena, zbog ograničene procesorske moći računara na kojima se nalaze.

11.4 Simulacija mobilnih mreža

Radi kompletnosti izlaganja, u najkraćim crtama ćemo se osvrnuti na osnove simulacije mobilnih mreža.

Odlika svih mobilnih mreža je da koriste radio-kanal kao sredinu za prenos. On je inherentno širokodifuzne prirode, što znači da slanje paketa iz jednog predajnog čvora rezultuje događajima prijema i obrade paketa u svim susednim čvorovima koji su u njegovom dometu, bez obzira na to kome je ovaj paket zaista namenjen. Iz istog razloga, pri simulaciji mobilnih mreža u obzir treba uzeti i sudare paketa.

Modeli radio-kanala su složeni. Treba posmatrati slabljenje putanje, refleksije, „zaklanjanje", feding, različite vrste interferencije itd. Zbog svega ovoga, ponekad je teško odrediti snagu signala na prijemu.

Zbog ograničenog dometa predajnika, u mobilnim mrežama se po pravilu koriste multihop *ad hoc* protokoli rutiranja.

Mobilnost korisnika, koji mogu biti krajnji sistemi ili čak čvorovi mreže, znači da se ovde menjaju liste suseda. Neki od često korištenih modela mobilnosti su:

- „slučajno kretanje", u kome se pretpostavlja da korisnik na slučajan način bira svoju novu lokaciju unutar oblasti i potom se slučajno odabranom brzinom pravolinijski kreće k njoj. Korisnik se u svakoj lokaciji zadržava slučajno vreme i potom ponavlja čitav proces iznova. Ovaj model je inspirisan Brownovim kretanjem čestica;
- kretanje po zadatoj putanji, u kome korisnik sledi predefinisanu putanju. Ovaj model dozvoljava korišćenje poznatih obrazaca mobilnosti, npr. putanja koje su snimljene u stvarnim mrežama;
- Manhattan model, u kome je kretanje dozvoljeno samo po nekim pravcima, koji odgovaraju ulicama.

Da bi se smanjio broj potrebnih izračunavanja i ovde se koriste različite heuristike. Na primer, ako je poznata maksimalna brzina čvora, može se odrediti vreme tokom koga se neće promeniti lista njegovih suseda.

Jedan od novijih modela mobilnosti, MARPL[1], ne daje predikciju položaja korisnika, već njegove povezanosti na baznu stanicu. Ovaj model je razvijen za potrebe optimizacije mobilnih mreža sa stanovišta potrošnje električne energije.

11.5 Simulator *ns*-3

Simulator *ns*-3 je slobodni softverski alat za simulaciju fiksnih i bežičnih telekomunikacionih mreža, namenjen za istraživačke i obrazovne svrhe. Projekat njegovog razvoja je pokrenut 2006. godine, s ciljem da se napravi potpuno novi simulator, koji bi bio

[1] Kateřina Dufková, Jean-Yves Le Boudec, Lukáš Kencl, and Milan Bjelica: "Predicting User-Cell Association in Cellular Networks from Tracked Data", *Lecture Notes in Computer Science* 5801, pp. 18–33, 2009.

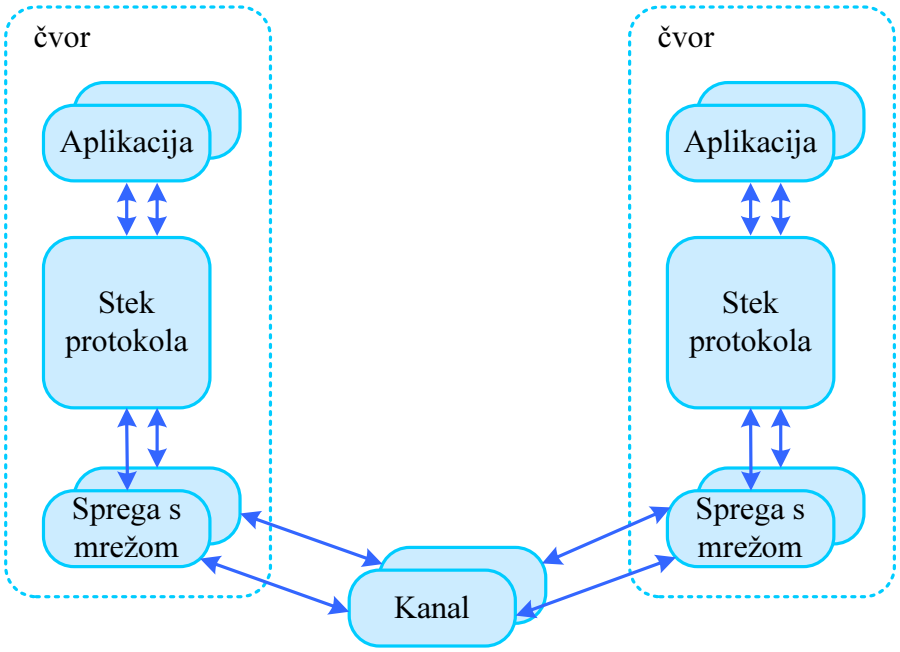
jednostavniji za korišćenje i čiji bi kôd bio konzistentniji od starog simulatora *ns*-2, koji je do tada široko korišćen u istraživačke svrhe.

Simulator *ns*-3 nije naredna niti unapređena verzija simulatora *ns*-2, već se radi o potpuno novom programu. Nije predviđena uzajamna kompatibilnost, pa se simulaciona skripta za *ns*-2 ne mogu direktno primeniti u *ns*-3.

Pitanje koje se često postavlja je koji od ova dva srodna simulatora treba izabrati za rad. Najbolje bi bilo da se nauče oba i da se odabere onaj koji je pogodniji za konkretnu primenu. Ukoliko to nije moguće, preporuka je da se novi projekti započinju u alatu koji se aktivnije razvija, a to je *ns*-3.

Preporučuje se instaliranje simulatora pod operativnim sistemom Linux. Programski kod je napisan u jeziku C++, dok se sama simulacija izvršava po principu simulacije diskretnih događaja.

Organizacija simulacionog modela u *ns*-3 prikazana je na slici 11.3.



Slika 11.3: Organizacija simulacionog modela u *ns*-3.

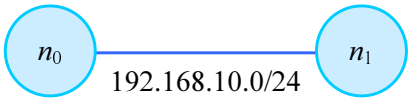
Elementi mreže se nazivaju čvorovima (*node*). Neki čvorovi odgovaraju mrežnim uređajima, kao što su to npr. ruteri, a neki krajnjim sistemima, poput umreženih računara. Ako čvorovi odgovaraju krajnjim sistemima, na njima se izvršavaju aplikacije (*application*). Na jednom čvoru se može izvršavati više aplikacija, a primeri onih koje su trenutno podržane su `BulkSendApplication`, `OnOffApplication`, `PacketSink`, `Ping6`, `Radvd`, `UdpEcho` i `UdpClientServer`. Aplikacije generišu i primaju saobraćaj, koji se posredstvom primenjenih protokola (npr. UDP, TCP, IPv6, IPv4, OSPF itd) prenosi ka sprezi s mrežom (mrežnom interfejsu – *net device*); nju možemo shvatiti kao mrežnu karticu na računaru. Sprega odgovara primenjenom kanalu (*channel*), pa tako postoje posebne sprege za kanale tipa tačka-tačka, CSMA, Aloha, WiFi, Wimax, LTE itd.

Jedan čvor se preko odgovarajućih sprega može istovremeno povezati na više kanala.

Iako je još u fazi razvoja, vidljivo je da je simulator *ns*-3 velik i složen sistem, koji nastoji da verno opiše procese u heterogenim mrežama. U odnosu na *ns*-2, ispravljene su uočene greške u implementiranju mehanizama protokola, a detaljnije su podržane bežične mreže. Pojedine funkcionalnosti su preuzete iz drugih simulatora, kao što je GTNetS. Negde se ide tako daleko da se npr. pri formiranju modela u obzir uzima i vrsta baterije u mobilnom terminalu.

Najveća zamerka koja se može izreći simulatoru *ns*-3 je da po svemu sudeći neće biti lak za učenje, čemu doprinosi i činjenica da se za sada simulacioni model ne formira u grafičkom interfejsu (jer ga nema), već putem skripata koje se pišu u jeziku C++ ili Pythonu. Uprkos tome, mnogi ugledni univerziteti su već uveli *ns*-3 u kurseve telekomunikacionih i računarskih mreža na osnovnim studijama, što treba shvatiti kao jaku preporuku za njegovo učenje. Naglasimo da neki vizuelizatori i animatori već postoje i sa njima ćemo se upoznati u odeljku 12.5. Takođe, zapisi rezultata mogu se obrađivati u matematičkim programima kao što su GNU Octave i gnuplot, ili specijalizovanim mrežnim alatima poput Wiresharka i tcpdumpa.

Kao primer korišćenja simulatora *ns*-3, posmatrajmo komunikaciju dvaju čvorova, prikazanih na slici 11.4.



Slika 11.4: Topologija posmatrane mreže.

Svaki čvor ima po jedan mrežni interfejs preko koga je povezan na link „tačka-tačka" kapaciteta 1 Mb/s i propagacionog kašnjenja 1 ms. Dodeljeni opseg IP adresa je 192.168.10.0/24. Na čvoru n_0 je instaliran UDP klijent, koji koristi port 10001, dok je na čvoru n_1 instaliran UDP server. Počevši od trenutka $t = 1$ s, od čvora n_0 se ka čvoru n_1 na svake dve sekunde šalju blokovi podataka veličine 512 B.

Pogledajmo kako bismo napisali odgovarajući simulacioni program u Pythonu. Počeli bismo pozivanjem modula koji su neophodni za izvršavanje simulacije:

```
import ns.core
import ns.network
import ns.point_to_point
import ns.internet
import ns.applications
```

Modul `core` je jezgro simulatora, dok ostali moduli redom omogućavaju formiranje čvorova, formiranje kanala, instaliranje protokolskog steka i instaliranje aplikacija.

Naredni korak je formiranje dvaju čvorova:

```
nodes = ns.network.NodeContainer()
nodes.Create(2)
```

posle čega formiramo link zadatih parametara

```
ptp = ns.point_to_point.PointToPointHelper()
ptp.SetDeviceAttribute("DataRate", ns.core.StringValue("1Mbps"))
ptp.SetChannelAttribute("Delay", ns.core.StringValue("1ms"))
```

i instaliramo sprege s mrežom

```
devices = ptp.Install(nodes)
```

Na svakom čvoru potom instaliramo IPv4 protokolski stek:

```
stack = ns.internet.InternetStackHelper()
stack.Install(nodes)
```

Sada svakom mrežnom interfejsu dodeljujemo adresu iz zadatog opsega:

```
address = ns.internet.Ipv4AddressHelper()
address.SetBase(ns.network.Ipv4Address("192.168.10.0"),
  ns.network.Ipv4Mask("255.255.255.0"))
interfaces = address.Assign (devices)
```

Adrese iz zadatog opsega se sukcesivno dodeljuju čvorovima, po redosledu kojim su pravljeni u grupi (`NodeContainer`).

Prelazimo na instaliranje UDP klijenta na čvoru n_0:

```
client = ns.applications.UdpClientHelper(interfaces.GetAddress(1), 10001)
client.SetAttribute("MaxPackets", ns.core.UintegerValue(3))
client.SetAttribute("Interval", ns.core.TimeValue(ns.core.Seconds (2.0)))
client.SetAttribute("PacketSize", ns.core.UintegerValue(512))
clientApps = client.Install(ns.network.NodeContainer(nodes.Get(0)))
clientApps.Start(ns.core.Seconds(1.0))
clientApps.Stop(ns.core.Seconds(7.0))
```

Primetimo da se kao argumenti metode `UdpClientHelper` redom zadaju IP adresa servera (odredišta) i port.

UDP server se instalira na čvoru n_1:

```
server = ns.applications.UdpServerHelper(10001)
serverApps = server.Install(ns.network.NodeContainer(nodes.Get(1)))
serverApps.Start(ns.core.Seconds(0.0))
serverApps.Stop(ns.core.Seconds(8.0))
```

Tekuće poruke o izvršavanju simulacije i međurezultati se mogu prikazivati na ekranu. Mi ćemo snimiti detaljnije izveštaje u *trace* i `pcap` datoteke, što je uobičajena praksa:

```
ascii = ns.network.AsciiTraceHelper()
ptp.EnableAsciiAll(ascii.CreateFileStream("primer.tr"))
ptp.EnablePcapAll ("primer")
```

Ostalo je još da zadamo početak simulacije i da po njenom završetku uništimo sve napravljene objekte i zatvorimo *trace* datoteku:

```
ns.core.Simulator.Run()
ns.core.Simulator.Destroy()
ascii.close()
```

Primetimo da je zatvaranje datoteke *lege artis* u Pythonu, mada se *ns*-3 na to buni.

Ako smo program npr. sačuvali kao datoteku `primer.py` u direktorijumu `ot4mis` unutar osnovnog direktorijuma simulatora, simulaciju ćemo pokrenuti iz osnovnog direktorijuma naredbom

```
./waf --pyrun ot4mis/primer.py
```

pri čemu će se *trace* i `pcap` datoteke sačuvati u osnovnom direktorijumu.

Simulaciona skripta je moguće pisati i u jeziku C++. Prethodni primer u njemu tako bi imao sledeći oblik:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

int main (int argc, char *argv[])
{
  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper ptp;
  ptp.SetDeviceAttribute ("DataRate", StringValue ("1Mbps"));
  ptp.SetChannelAttribute ("Delay", StringValue ("1ms"));

  NetDeviceContainer devices;
  devices = ptp.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("192.168.10.0", "255.255.255.0");
```

```
  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpClientHelper client (interfaces.GetAddress (1), 10001);
  client.SetAttribute ("MaxPackets", UintegerValue (3));
  client.SetAttribute ("Interval", TimeValue (Seconds (2.0)));
  client.SetAttribute ("PacketSize", UintegerValue (512));

  ApplicationContainer clientApps = client.Install (nodes.Get (0));
  clientApps.Start (Seconds (1.0));
  clientApps.Stop (Seconds (7.0));

  UdpServerHelper server (10001);

  ApplicationContainer serverApps = server.Install (nodes.Get (1));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (Seconds (8.0));


  AsciiTraceHelper ascii;
  ptp.EnableAsciiAll (ascii.CreateFileStream ("primer.tr"));
  ptp.EnablePcapAll ("primer");

  Simulator::Run ();
  Simulator::Destroy ();
}
```

Preporuka je da se C++ programi čuvaju u direktorijumu `scratch`. Naš program bi se u tome slučaju pokretao naredbom `./waf --run scratch/primer`.

Simulacioni model je prilično jednostavan, dok su programi u oba slučaja dugački. Simulator, naime, zahteva neke opšte deklaracije (učitavanje modula, zapis rezultata i sl) koje „potroše" mnogo linija kôda; ovaj svojevrsni ofset je manje uočljiv u slučaju komplikovanijih mrežnih topologija.

Razmotrimo sada analizu rezultata koje dobijamo po izvršavanju simulacije. Datoteka *trace* sadrži detaljne zapise događaja u predajnom baferu svakog mrežnog interfejsa u simulaciji; svakom događaju odgovara jedan red u datoteci. Prva tri reda zapisa, koje radi preglednosti prikazujemo u formatiranom obliku, glase ovako:

```
+
1
/NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue
ns3::PppHeader (
   Point-to-Point Protocol: IP (0x0021))
   ns3::Ipv4Header (
      tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes)
      0 flags [none] length: 540 192.168.10.1 > 192.168.10.2)
      ns3::UdpHeader (
         length: 520 49153 > 10001)
```

```
         ns3::SeqTsHeader (
            (seq=0 time=1))
            Payload (size=500)

-
1
/NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue
ns3::PppHeader (
   Point-to-Point Protocol: IP (0x0021))
   ns3::Ipv4Header (
      tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes)
      0 flags [none] length: 540 192.168.10.1 > 192.168.10.2)
      ns3::UdpHeader (
         length: 520 49153 > 10001)
         ns3::SeqTsHeader (
            (seq=0 time=1))
            Payload (size=500)

r
1.00534
/NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/MacRx
ns3::Ipv4Header (
   tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes)
   0 flags [none] length: 540 192.168.10.1 > 192.168.10.2)
   ns3::UdpHeader (
      length: 520 49153 > 10001)
      ns3::SeqTsHeader (
         (seq=0 time=1))
         Payload (size=500)
```

Prva oznaka u redu se odnosi na vrstu događaja – pristizanje paketa u bafer (+), izlazak paketa iz bafera (-), odbacivanje paketa ukoliko je bafer bio popunjen (d), ili prijem paketa na odredištu (r). Potom slede simulaciono vreme (u sekundama), opis objekta koji je izvor događaja i opis primenjenih protokola. U prikazanom primeru, vidimo da je paket došao u trenutku 1 u predajni bafer mrežnog interfejsa tipa „tačka-tačka" s rednim brojem 0 na čvoru 0. Unutar ppp (*point to point protocol*) okvira, enkapsuliran je IPv4 paket veličine 540 B, od izvora čija je adresa 192.168.10.1 ka odredištu 192.168.10.2. Kad oduzmemo 20 B zaglavlja, preostaje 520 B korisnog sadržaja koji predstavljaju UDP paket od izvorišnog porta 49153, koji nije eksplicitno zadat, već ga je simulator tako interno numerisao, ka portu 10001. Zanimljivo je primetiti da je zaglavlje UDP paketa veličine 8 B, što bi se uklopilo u zadatih 512 B korisnog sadržaja; ipak, *ns*-3 nas na kraju obaveštava da je korisni sadržaj veličine 500 B, jer aplikacija (`UdpClientServer`) u svakom paketu alocira 4 B za oznaku sekvence i 8 B za oznaku vremena.

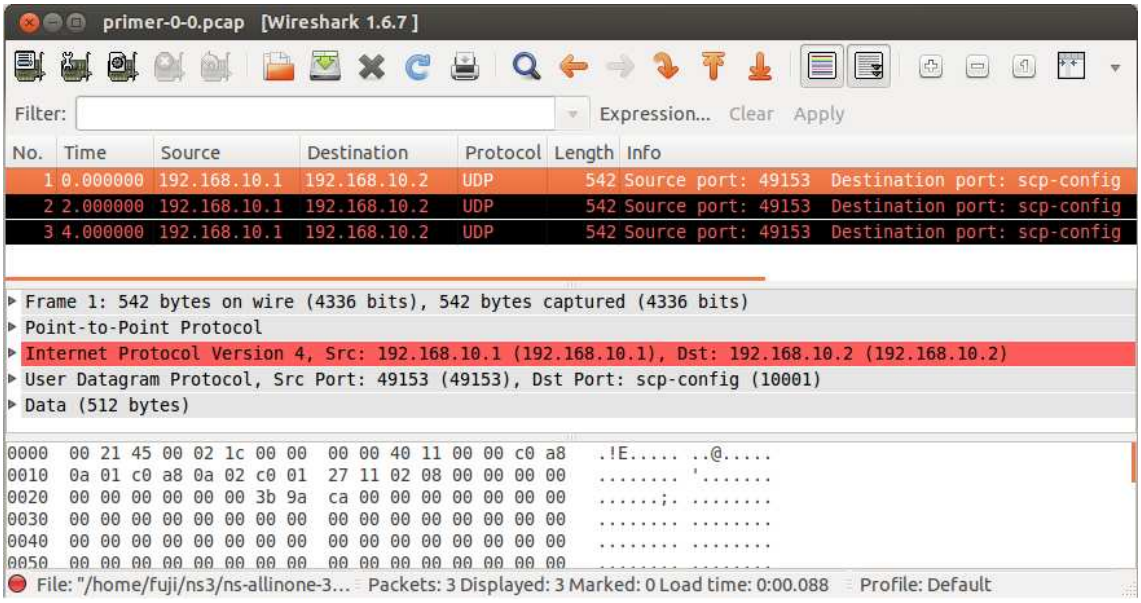
U drugoj liniji zapisa vidimo da je u trenutku 1 s ovaj paket napustio predajni bafer mrežnog interfejsa s rednim brojem 0 na čvoru 0.

Treća linija nas obaveštava da je paket primljen u trenutku 1,00534 s. Proverimo da

li je ovo ispravan rezultat. Paket je poslat u trenutku 1 s. Propagaciono kašnjenje na linku je 1 ms. Ukupna dužina paketa je 542 B (540 B za IPv4 paket i 2 B za ppp zaglavlje), pa je potrebno 4,336 ms da bi se on „utisnuo" u link protoka 1 Mb/s. To znači da paket stiže na odredište u trenutku 1,005336 s, što uz grešku zaokruživanja odgovara prikazanoj vrednosti 1,00534 s.

Na sličan način su opisani slanje i prijem i narednih dvaju paketa, pa ih ovde nećemo detaljno analizirati.

Pošto smo zadali snimanje izveštaja i u vidu `pcap` datoteka, za svaki link će se formirati datoteka `naziv_programa-broj_čvora-broj_intefrejsa.pcap`. Ove datoteke ćemo obraditi u programu Wireshark, čiji je glavni prozor prikazan na slici 11.5.



Slika 11.5: Izgled glavnog prozora programa Wireshark.

U primeru sa slike, prikazan je saobraćaj na (jedinom) mrežnom interfejsu čvora n_0. Za svaki paket, prikazani su trenutak generisanja (ili, u opštem slučaju, prijema), IP adrese izvorišta i odredišta, primenjeni transportni protokol, veličina paketa i dodatni podaci (npr. izvorišni i odredišni port). Treba naglasiti da Wireshark meri vreme u odnosu na prvi događaj (u ovom slučaju, slanje paketa broj 1), a ne na simulaciono vreme u *ns*-3.

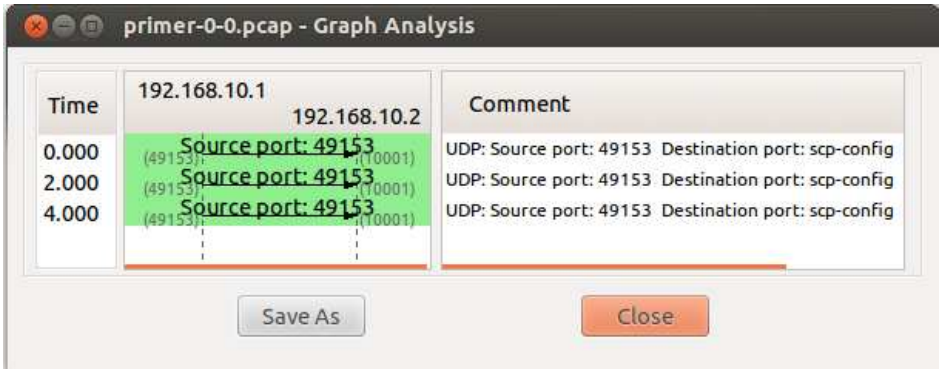
Izborom odgovarajućih opcija, za svaki paket je moguće dobiti detaljne podatke, sve do vrednosti pojedinih bita u njemu. Primer ovakvog izveštaja za prvi paket dat je u nastavku.

```
No.    Time        Source          Destination     Protocol  Length
  1    0.000000    192.168.10.1    192.168.10.2    UDP       542
 Info
 Source port: 49153   Destination port: scp-config

Frame 1: 542 bytes on wire (4336 bits), 542 bytes captured (4336 bits)
```

```
    Arrival Time: Jan  1, 1970 01:00:01.000000000 Central Europe Standard
 Time
    Epoch Time: 1.000000000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 542 bytes (4336 bits)
    Capture Length: 542 bytes (4336 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: ppp:ip:udp:data]
    [Coloring Rule Name: Checksum Errors]
    [Coloring Rule String: cdp.checksum_bad==1 || edp.checksum_bad==1 ||
 ip.checksum_bad==1 || tcp.checksum_bad==1 || udp.checksum_bad==1 ||
 sctp.checksum_bad==1 || mstp.checksum_bad==1]
Point-to-Point Protocol
    Protocol: IP (0x0021)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst:
 192.168.10.2 (192.168.10.2)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00:
 Not-ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-
 Capable Transport) (0x00)
    Total Length: 540
    Identification: 0x0000 (0)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x0000 [incorrect, should be 0xe37d (maybe caused by "IP
 checksum offload"?)]
    Source: 192.168.10.1 (192.168.10.1)
    Destination: 192.168.10.2 (192.168.10.2)
User Datagram Protocol, Src Port: 49153 (49153), Dst Port: scp-config (10001)
    Source port: 49153 (49153)
    Destination port: scp-config (10001)
    Length: 520
    Checksum: 0x0000 (none)
Data (512 bytes)
```

Opcijom `IO Graphs` iz menija `Statistics` mogu se nacrtati grafici saobraćaja na posmatranom interfejsu, dok se opcijom `Flow Graph` iz istog menija dobija grafički prikaz toka razmene paketa (slika 11.6).

Prikazane podatke je moguće sačuvati u nekoliko formata – `txt`, `csv` ili `ps` i potom



Slika 11.6: Dijagram razmene paketa u mreži.

uvesti u specijalizovane matematičke ili grafičke programe radi dalje obrade.

Poglavlje 12

Primeri simulacije u telekomunikacijama

U ovome poglavlju je dato nekoliko tipičnih primera simulacije telekomunikacionih sistema.

12.1 Sistem za prenos signala postupkom 16-QAM

12.1.1 Zadatak

Naš prvi zadatak je tipičan „školski": Treba simulirati sistem za prenos signala postupkom 16-QAM kanalom u kome deluju feding i beli šum.

12.1.2 Opis modela

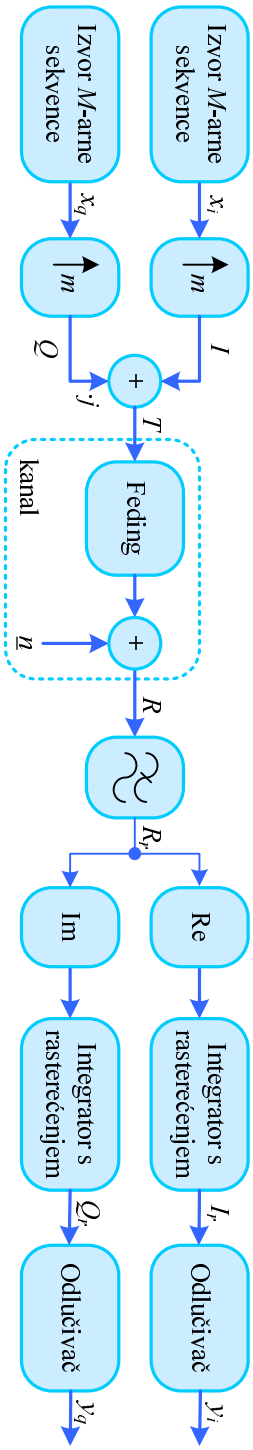
Da bismo smanjili količinu podataka koje treba obraditi, odlučili smo se za analizu sistema preko niskofrekvencijskog ekvivalenta. Blok-šema simulacionog modela prikazana je na slici 12.1.

Niskofrekvencijski ekvivalent 16-QAM signala je

$$T = I + jQ, \tag{12.1}$$

gde su sa I i Q redom označeni kvaternarni signali u fazi i kvadraturi, posle izvršenog nadodabiranja.

Pretpostavili smo da u kanalu deluje Rayleighjev feding, pa je signal na ulazu prije-



Slika 12.1: Blok-šema simulacionog modela.

mnika

$$R = T(\xi + F\exp(-j\Phi)) + \underline{n}. \tag{12.2}$$

U gornjem izrazu, faktor ξ opisuje oslabljenu direktnu komponentu signala, F je slučajna promenljiva s Rayleighjevom raspodelom koja opisuje fluktuaciju amplitude usled fedinga, dok je Φ slučajna promenljiva uniformno raspodeljena na intervalu $[0, 2\pi)$, koja opisuje fluktuaciju faze. Sa $\underline{n}$ je označen kompleksni Gaussov šum,

$$\underline{n} = n_i + jn_q. \tag{12.3}$$

Na ulazu prijemnika se nalazi Butterworthov filtar. Potom se razdvajaju realni i kompleksni deo signala, na njih se primenjuje integriranje s rasterećenjem i donosi se odluka o primljenom simbolu.

12.1.3 Opis simulacije

Simulacija je realizovana u programu GNU Octave. Rayleighjeva slučajna promenljiva s parametrom σ generisana je transformacionom metodom, tako što je prvo generisan slučajan broj U iz raspodele $\mathrm{Unif}\,(0, 1)$, a potom je izvršena transformacija

$$F = \sigma\sqrt{-2\ln(1-U)}. \tag{12.4}$$

Ostali elementi simulacije su analogni primeru iz odeljka 5.1, s tim što se sad posmatraju dve grane signala.

Listing programskog koda dat je u nastavku.

```
% zadavanje vrednosti parametara
N = 5e6;
m = 5;
sigma = 5e-5;
s = 1e-3;
ksi = 1e-3;

% sekvence u fazi i kvadraturi
xi = randint(1, N, [0, 3])*2 - 3;
xq = randint(1, N, [0, 3])*2 - 3;

% nadodabiranje
I = [];
Q = [];

for cnt = 1:N
  I = [I, xi(cnt)*ones(1, m)];
  Q = [Q, xq(cnt)*ones(1, m)];
endfor

% izlaz predajnika
```

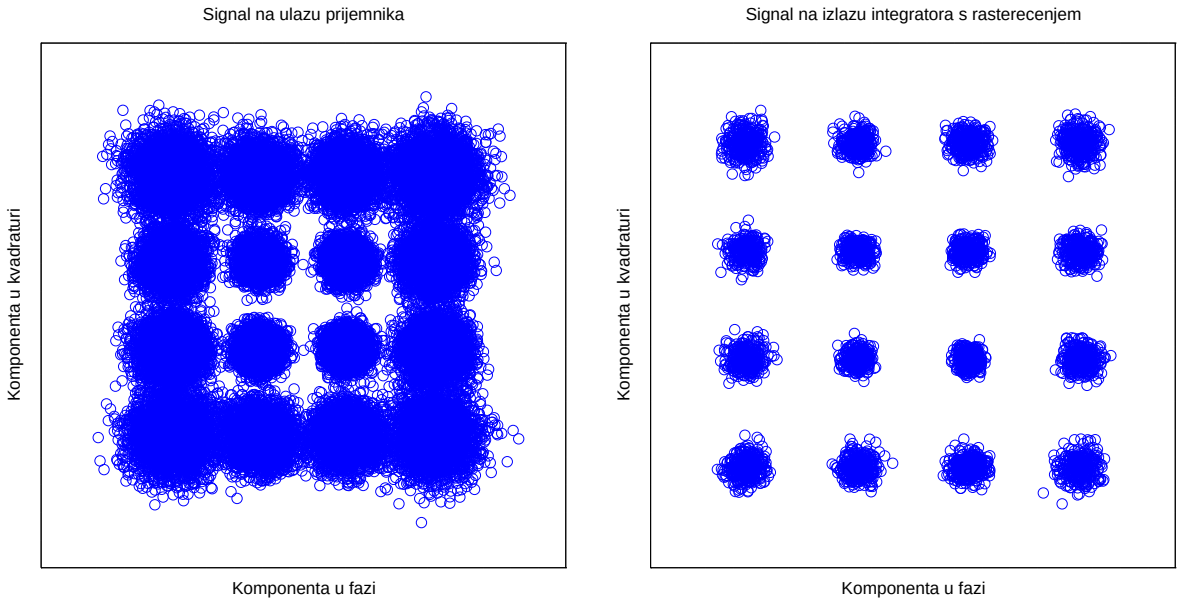
```
T = I + i*Q;

% Rayleighjev feding
F = sqrt(-2*log(1 - rand(1, N*m))).*sigma;
phi = unifrnd(0, 2*pi, 1, N*m);

% kompleksni uskopojasni Gaussov šum
ni = s*randn(1, N*m);
nq = s*randn(1, N*m);
n = ni + i*nq;

% ulaz prijemnika
R = T.*(ksi + F.*exp(-i*phi)) + n;
scatterplot(R, 1, 0, 'bo')
title('Konstelacioni dijagram signala na ulazu prijemnika')
xlabel('Komponenta u fazi'), ylabel('Komponenta u kvadraturi')

% ulazni filtar
% Butterworth NF, 4. reda
[b, a] = butter(4, 0.78);

% signal na izlazu filtra
Rr = filter(b, a, R);

% integratori s rasterećenjem
Ir = [];
Qr = [];
for cnt = 1:N
  Ir = [Ir, mean(real(Rr((cnt-1)*m+1:cnt*m)))];
  Qr = [Qr, mean(imag(Rr((cnt-1)*m+1:cnt*m)))];
endfor
scatterplot(Ir + i*Qr, 1, 0, 'bo')
title('Konstelacioni dijagram signala na izlazu integratora s rasterecenjem')
xlabel('Komponenta u fazi'), ylabel('Komponenta u kvadraturi')

yi = [];
yq = [];

% odlučivanje
vi = (max(Ir) - min(Ir))/6;
vq = (max(Qr) - min(Qr))/6;
Ir = Ir/vi;
Qr = Qr/vq;

for cnt = 1:N
  if (Ir(cnt) <= -2)
    yi(cnt) = -3;
  elseif ((-2 < Ir(cnt)) & (Ir(cnt) <= 0))
    yi(cnt) = -1;
  elseif ((0 < Ir(cnt)) & (Ir(cnt) <= 2))
```

```
    yi(cnt) = 1;
  else
    yi(cnt) = 3;
  endif
  if (Qr(cnt) <= -2)
    yq(cnt) = -3;
  elseif ((-2 < Qr(cnt)) & (Qr(cnt) <= 0))
    yq(cnt) = -1;
  elseif ((0 < Qr(cnt)) & (Qr(cnt) <= 2))
    yq(cnt) = 1;
  else
    yq(cnt) = 3;
  endif
endfor

SER = (sum(yi!=xi) + sum(yq!=xq))/(2*N)
```

12.1.4 Rezultati

Za brojčane vrednosti parametara kao u listingu, ocenjena verovatnoća greške simbola je iznosila $4{,}3 \cdot 10^{-4}$. Kada smo ukinuli feding, dobili smo verovatnoću greške približno jednaku nuli.

Konstelacioni dijagrami signala na ulazu prijemnika i na izlazu integratora s rastereće-njem, u slučaju kada deluje feding, dati su na slici 12.2. Njihovim poređenjem se jasno uočava značaj prilagođenog filtra u prijemniku.



Slika 12.2: Konstelacioni dijagrami signala.

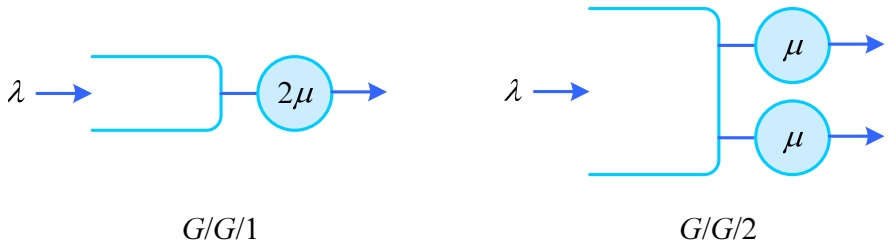
12.2 Poređenje servisnih sistema

12.2.1 Zadatak

Dva čvora brze Ethernet mreže mogu se povezati jednim linkom kapaciteta $2C$ ili dvama linkovima koji imaju zajednički bafer i čiji kapaciteti iznose $C = 100$ Mb/s. Uz pretpostavku da je generisanje okvira opisano pomerenom eksponencijalnom raspodelom, a da njihova dužina ima ograničenu Pareto raspodelu, čije je matematičko očekivanje $\mathrm{E}X = 1000$ b, koje rešenje je bolje sa stanovišta prosečnog kašnjenja okvira?

12.2.2 Opis modela

Opisane konfiguracije odgovaraju redom servisnim sistemima $G/G/1$, čiji je protok dolazaka λ i protok obrade 2μ i $G/G/2$, čiji je protok dolazaka λ i protok obrade μ.



Slika 12.3: Razmatrane konfiguracije servisnih sistema.

Vreme međudolazaka okvira ima pomerenu eksponencijalnu raspodelu, čija je funkcija gustine verovatnoće

$$f_T(t) = \begin{cases} 0, & t < a \\ b\exp[-b(t-a)], & t \geqslant a \end{cases}. \tag{12.5}$$

Tzv. parametar položaja a odgovara minimalnom vremenu međudolazaka okvira (vremenu između početaka susednih okvira), koje je za Ethernet jednako zbiru trajanja najkraćeg okvira ($L = 72$ B) i rastojanja između okvira (*interframe gap* od 12 B). Vrednost parametra oblika b ćemo varirati da bismo opisali promenu protoka okvira.

Dužina okvira ima ograničenu Pareto raspodelu, čija je funkcija gustine verovatnoće

$$f_X(x) = \frac{\alpha L^\alpha x^{-\alpha-1}}{1-\left(\frac{L}{H}\right)^\alpha}, \quad L \leqslant x \leqslant H. \tag{12.6}$$

Pri tome je $H = 1526$ B dužina najdužeg Ethernet okvira.

Za $\alpha \neq 1$, matematičko očekivanje ove raspodele je

$$\mathrm{E}X = \frac{L^\alpha}{1-\left(\frac{L}{H}\right)^\alpha}\frac{\alpha}{\alpha-1}\left(\frac{1}{L^{\alpha-1}} - \frac{1}{H^{\alpha-1}}\right). \tag{12.7}$$

Numeričkim rešavanjem, dobijamo da zadatoj vrednosti $\mathrm{E}X = 1000$ b odgovara vrednost parametra $\alpha = 2{,}3$.

12.2.3 Opis simulacije

Primenićemo Monte Carlo simulaciju, u kojoj ćemo alternativne konfiguracije sistema uporediti pod ekvivalentnim uslovima.

Slučajne brojeve ćemo generisati na sledeći način. Raspodelu (12.5) ćemo generisati tako što ćemo slučajnu promenljivu $Q \sim \mathrm{Exp}(b)$ sabrati s konstantom a. Slučajnu promenljivu datu sa (12.6) ćemo generisati indirektnom metodom. Prvo ćemo generisati slučajnu promenljivu $U \sim \mathrm{Unif}\,(0,1)$, a zatim izvršiti transformaciju

$$X = \left(\frac{UL^\alpha - UH^\alpha + H^\alpha}{H^\alpha L^\alpha} \right)^{-\frac{1}{\alpha}}. \tag{12.8}$$

Može se pokazati da X ima traženu ograničenu Pareto raspodelu. Konačno, trajanje obrade okvira će biti

$$W = \frac{X}{C}, \tag{12.9}$$

gde je $C = 100$ Mb/s. Dvostruko veći protok obrade sistema $G/G/1$ ostvarićemo tako što ćemo generisanu sekvencu s količinom posla, W, podeliti sa dva.

Podsetimo se sada simulacionog modela servisnog sistema $M/M/1$, koji smo razmotrili u odeljku 3.8, na strani 37. S izuzetkom generisanja slučajnih brojeva, sistem $G/G/1$ ćemo simulirati na analogan način, dok ćemo kod sistema $G/G/2$ posmatrati dva pokazivača odlazaka, j_1 i j_2, koji odgovaraju svakom od servisera.

				$i \downarrow$	
	1	2	3	4	$\cdots$
A	0,1	0,5	0,7	1,1	$\cdots$
W	0,2	0,8	0,6	0,7	$\cdots$
D	0,3	1,3	1,3		
	1	2	3	4	$\cdots$
		$\uparrow$	$\uparrow$		
		j_2	j_1		

Slika 12.4: Nizovi vremena i pokazivači događaja u servisnom sistemu $G/G/2$.

```
# queues.py
# poređenje servisnih sistema u Pythonu
```

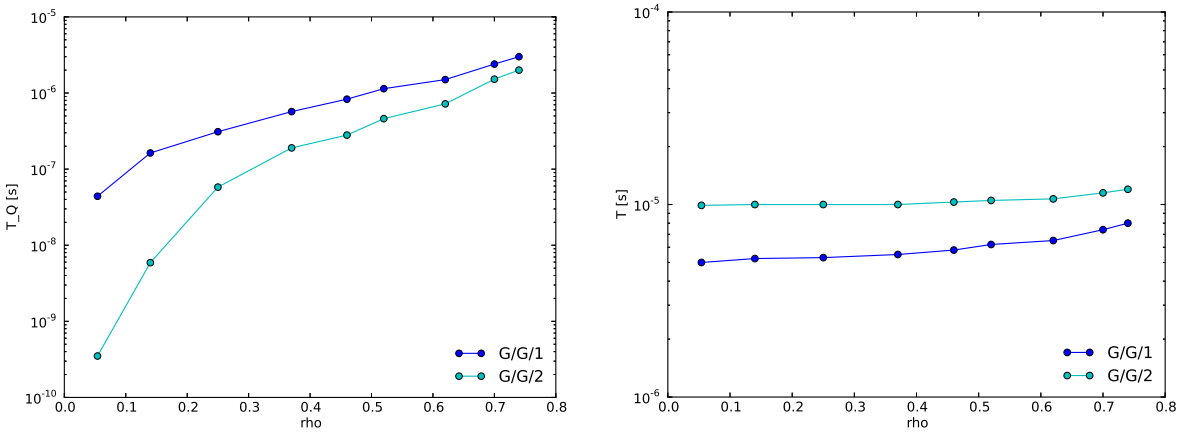
```
# poziv funkcije: python queues.py lambd k

from scipy import *
import random, sys

lambd = float(sys.argv[1])
k = int(sys.argv[2])

# inicijalizacija
n = int(1.2*k)
IA = []
W = []
A = []
L = 72.*8
H = 1526.*8
alpha = 2.3

# generisanje slučajnih promenljivih
for m in range(n):
   IA.append(random.expovariate(lambd) + (12+72)*8/100e6)
   U = random.uniform(0, 1)
   P = ((-U*H**alpha + U*L**alpha + H**alpha)/(H*L)**alpha)**(-1/alpha)
   W.append(P/100e6)
   A.append(sum(IA[0:m+1]))

W = array(W)

# sistem G/G/1
i_a = 0
j_a = 0
W_a = W/2
D_a = zeros(n)
t_Q_a = zeros(n)
t_a = zeros(n)
D_a[0] = A[0] + W_a[0]
i_a += 1
while (j_a < k):
   if (A[i_a] < D_a[j_a]):
      # dolazak
      i_a += 1
   else:
      # odlazak
      t_Q_a[j_a] = D_a[j_a] - A[j_a] - W_a[j_a]
      t_a[j_a] = D_a[j_a] - A[j_a]
      j_a += 1
      if (j_a != i_a):
         # serviser je zauzet
         D_a[j_a] = D_a[j_a-1] + W_a[j_a]
      else:
         # serviser je slobodan
```

```
         D_a[j_a] = A[j_a] + W_a[j_a]

# ocena i prikaz rezultata
t_stop_a = D_a[k-1]
rho_a = sum(W_a[0:k])/t_stop_a
T_a = sum(t_a)/k
T_Q_a = sum(t_Q_a)/k

disp("Sistem G/G/1:")
disp([rho_a, T_Q_a, T_a])

# sistem G/G/2
i_b = 0
j1_b = 0; j2_b = 1
W_b = W
D_b = zeros(n)
t_Q_b = zeros(n)
t_b = zeros(n)
D_b[0] = A[0] + W_b[0]
D_b[1] = A[1] + W_b[1]
i_b += 1
while ((j1_b < k) and (j2_b < k)):
   if ((A[i_b] < D_b[j1_b]) and (A[i_b] < D_b[j2_b])):
      # dolazak
      i_b += 1
   else:
      # odlazak
      if (D_b[j1_b] < D_b[j2_b]):
         # odlazak iz prve radionice
         t_Q_b[j1_b] = D_b[j1_b] - A[j1_b] - W_b[j1_b]
         t_b[j1_b] = D_b[j1_b] - A[j1_b]
         if (max(j1_b, j2_b) + 1 >= i_b):
            # prvi serviser je slobodan
            j1_b = max(j1_b, j2_b) + 1
            D_b[j1_b] = A[j1_b] + W_b[j1_b]
         else:
            # prvi serviser je zauzet
            aux = D_b[j1_b]
            j1_b = max(j1_b, j2_b) + 1
            D_b[j1_b] = aux + W_b[j1_b]
      else:
         # odlazak iz druge radionice
         t_Q_b[j2_b] = D_b[j2_b] - A[j2_b] - W_b[j2_b]
         t_b[j2_b] = D_b[j2_b] - A[j2_b]
         if (max(j1_b, j2_b) + 1 >= i_b):
            # drugi serviser je slobodan
            j2_b = max(j1_b, j2_b) + 1
            D_b[j2_b] = A[j2_b] + W_b[j2_b]
         else:
            # drugi serviser je zauzet
```

```
            aux = D_b[j2_b]
            j2_b = max(j1_b, j2_b) + 1
            D_b[j2_b] = aux + W_b[j2_b]

# ocena i prikaz rezultata
t_stop_b = D_b[k-1]
rho_b = sum(W_b[0:k])/(2*t_stop_b)
T_b = sum(t_b)/k
T_Q_b = sum(t_Q_b)/k

disp("Sistem G/G/2:")
disp([rho_b, T_Q_b, T_b])
```

12.2.4 Rezultati

Rezultati simulacije su prikazani na slici 12.5. Vidimo da je sistem $G/G/2$ bolji po pitanju zadržavanja okvira u baferu, dok je sistem $G/G/1$ bolji po pitanju ukupnog kašnjenja. Pošto je dominantno kašnjenje obrade (pri „utiskivanju" okvira u link), ukupno kašnjenje slabo zavisi od iskorišćenosti servisera, tj. od protoka okvira, naročito u slučaju sistema $G/G/1$.



Slika 12.5: Prosečno zadržavanje okvira u baferu (levo) i prosečno kašnjenje okvira (desno).

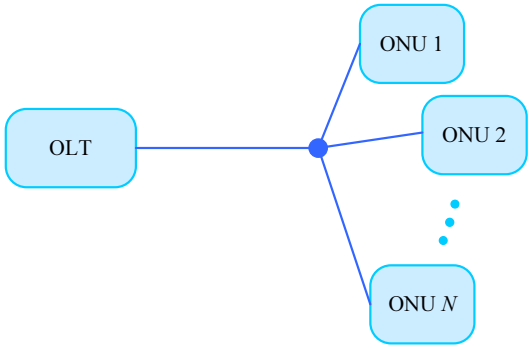
12.3 Procedura registracije u mrežama IEEE 802.3av

12.3.1 Uvod

Standard IEEE 802.3av opisuje pasivne optičke telekomunikacione mreže koje nude protok do 10 Gb/s u oba smera prenosa. Zbog visokog protoka i popularnosti Ether-

net tehnologije na kojoj se zasnivaju, ove mreže bi u skorijoj budućnosti mogle naći značajnu primenu u realizaciji telekomunikacione pristupne infrastrukture.

Struktura ovakve mreže prikazana je na slici 12.6. Radi se o jednostavnoj topologiji stabla, čiji koren predstavlja optički linijski terminal (OLT) na lokaciji operatora, dok listove predstavljaju optičke mrežne jedinice (ONU) na lokacijama korisnika. U podstanici, koja predstavlja tačku razgranavanja mreže, smešten je pasivni optički razdelnik/sabirač snage. Optičko vlakno koje spaja OLT s tačkom razgranavanja naziva se „feeder", dok se vlakna kojima se tačka razgranavanja spaja sa ONU uređajima nazivaju „drop".



Slika 12.6: Topologija mreže prema standardu IEEE 802.3av.

ONU uređaji predstavljaju korisničke stanice u mreži. Da bi se omogućila delimična kompatibilnost sa starim standardom 802.3ah, standard 802.3av podržava tri vrste optičkih mrežnih jedinica – „stare", s protokom od 1 Gb/s u oba smera; „mešane", s protokom od 1 Gb/s u smeru naviše i 10 Gb/s u smeru naniže i „nove", s protokom od 10 Gb/s u oba smera.

U nastavku ćemo razmotriti računarsku simulaciju čiji je cilj sagledavanje performansi mreže 802.3av tokom postupka pronalaženja i registrovanja novopriključenih optičkih mrežnih jedinica na optički linijski terminal; ove aktivnosti se izvršavaju na podsloju kontrole pristupa sredini za prenos (MAC), unutar sloja linka podataka (DLL).

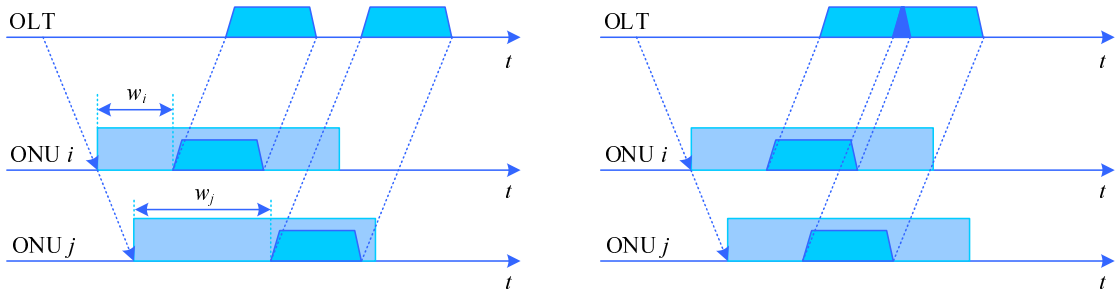
12.3.2 Opis modela

Postupak registracije odgovara standardnoj proceduri „rukovanja" koja se odvija na sledeći način. OLT periodično otvara tzv. prozor pronalaženja, tokom koga se novi ONU uređaji mogu prijaviti i registrovati. Početak ovoga prozora najavljuje se porukom GATE. ONU uređaji koji žele da se registruju čekaju slučajno vreme po otpočinjanju prozora pronalaženja i potom odgovaraju porukom REGISTER_REQ. OLT potvrđuje registraciju slanjem poruke REGISTER i, konačno, još jednom porukom GATE obaveštava ONU o dodeljenom vremenskom slotu za pristup uzlaznom linku. ONU potvrđuje njen prijem slanjem poruke REGISTER_ACK.

ONU uređaji ne mogu uzajamno neposredno komunicirati, pa je moguće da će više

njih istovremeno pokušati da se registruje. U tom slučaju, doći će do sudara njihovih poruka REGISTER_REQ i registracija neće biti uspešna. Ovakvi ONU čekaju da OLT otvori novi prozor pronalaženja i potom ponovo pokušavaju da se registruju. Proces registracije se stoga inherentno odlikuje nadmetanjem za zajednički resurs, pa se njegove performanse mogu posmatrati kroz verovatnoću sudara ili uspešne registracije.

Primeri uspešne i neuspešne registracije ONU uređaja i i j ilustrovani su na slici 12.7. Osenčenim pravougaonicima označeni su prozori pronalaženja, dok su poruke REGISTER_REQ s pripadajućim zaštitnim vremenskim intervalima označene trapezima.



Slika 12.7: Primeri uspešne (levo) i neuspešne registracije dvaju ONU uređaja (desno).

Do sudara neće doći ukoliko bude važio uslov

$$|(t_i + w_i) - (t_j + w_j)| \leqslant T, \tag{12.10}$$

gde su sa t_i i t_j označena pripadajuća kašnjenja potrebna za prenos u oba smera; w_i i w_j su slučajna kašnjenja unutar prozora pronalaženja, dok je T ukupno vreme potrebno za slanje poruke REGISTER_REQ, sa svim zaštitnim vremenskim intervalima.

U mreži sa N ONU uređaja, $N \geqslant 2$, uslov (12.10) važi za svaki uočeni par (i, j).

Odredimo sada vrednost T. Ona mora da obuhvati trajanje Ethernet okvira, u čijem je polju korisnog sadržaja poruka REGISTER_REQ, kao i dodatne zaštitne intervale koje standard propisuje da bi se završili prelazni procesi na predajnoj i prijemnoj strani:

$$\begin{aligned} T =& T_{on} + T_{receiver\_settling} + T_{CDR} + \\ &+ T_{code\_group\_align} + |\text{PREAMBLE}| + |\text{SoF}| + \\ &+ |\text{REGISTER\_REQ}| + T_{off} + |\text{IPG}|\,. \end{aligned} \tag{12.11}$$

Vrednosti pojedinačnih komponenti su date u tabeli 12.1. U najnepovoljnijem slučaju se dobija $T = 2{,}2912$ μs.

12.3.3 Opis simulacije

U nizu Monte Carlo simulacija koje su izvršene u programskom paketu Python, ocenjena je verovatnoća uspešne registracije ONU uređaja u prvom pokušaju, P_{OK}, u

Tabela 12.1: Komponente vremena potrebnog za prenos poruke REGISTER_REQ.

Oznaka	Vrednost	Referenca u standardu
T_{on}	< 512 ns	60.7.13.1.1, Table 75-8,9
$T_{receiver\_settling}$	800 ns	60.7.13.2.1, Table 75-6,7
T_{CDR}	< 400 ns	76.4.2.1
$T_{code\_group\_align}$	0	36.3.2.4, 75.7.14
|PREAMBLE|	5.6 ns (7 B)	3.2.1
|SoF|	0.8 ns (1 B)	3.2.2
|REGISTER_REQ|	51.2 ns (64 B)	77.3.6.3
T_{off}	< 512 ns	60.7.13.11.1, Table 75-8,9
|IPG|	9.6 ns (96 b)	Table 4.2

mreži sa $N \in [2, 64]$ ONU. Nezavisne replikacije su ponavljane 10^5 puta, za šta je probom ustanovljeno da s verovatnoćom od 90% daje relativnu grešku ocene ne veću od 2%.

Posmatran je najgori slučaj, koji nastupa kada su ONU uređaji ekvidistantno raspoređeni u odnosu na OLT. Tada je $t_i = t_j$ za svaki par (i, j). Bez umanjenja opštosti, možemo pretpostaviti da je vrednost propagacionog kašnjenja jednaka nuli, pa se uslov (12.10) svodi na

$$|w_i - w_j| \leqslant T. \tag{12.12}$$

Vrednost slučajnog kašnjenja unutar prozora pronalaženja je uniformno raspodeljena na intervalu $[0, w_{max}]$, pri čemu je $w_{max} \in [1, 500]$ μs. Minimalno trajanje prozora pronalaženja tada je $t_{max} + w_{max} + T$.

Ocenjena verovatnoća uspešne registracije, P_{OK}, potom je iskorišćena da bi se proračunala efikasnost registracije U, koja predstavlja odnos vremena koje je efektivno potrebno za prenos poruka REGISTER_REQ koje se nisu sudarile i trajanja prozora pronalaženja:

$$U = \frac{T}{t_{max} + w_{max} + T} N P_{OK}. \tag{12.13}$$

U prethodnim izrazima, t_{max} je maksimalno kašnjenje potrebno za prenos u oba smera, za koje smo pretpostavili da je jednako nuli.

Programski kod dat je u nastavku.

```
# 802.py
# simulacija mreže IEEE 802.3av u Pythonu
# poziv funkcije: python 802.py N Tw_max
# N - broj ONU
# Tw_max - maksimalno slučajno kašnjenje

from pylab import *
```

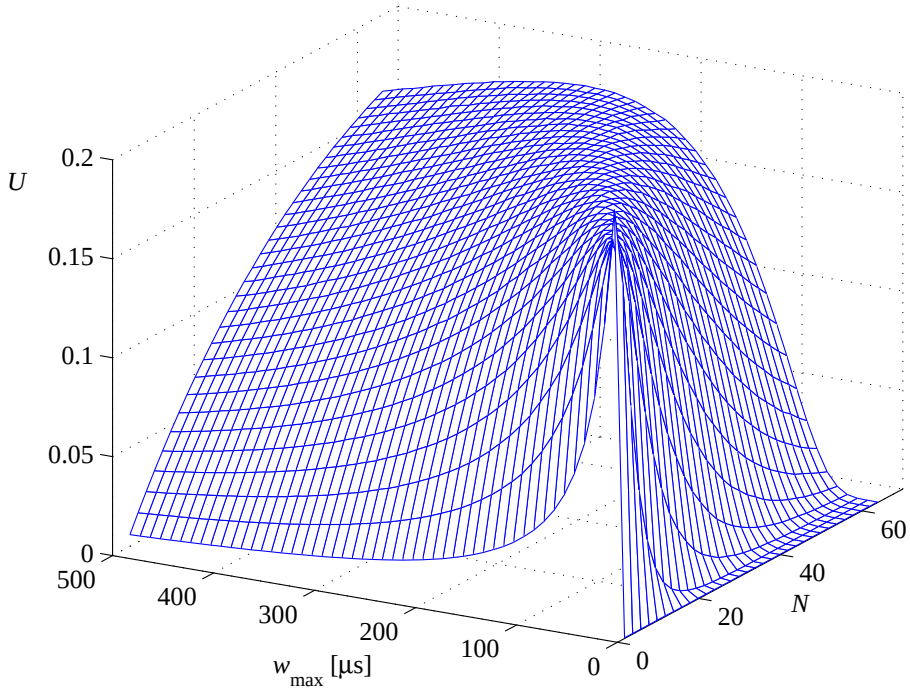
```
N = int(sys.argv[1])
Tw_max = float(sys.argv[2])

T = 2.2912e-6 # trajanje prozora pronalaženja
k = 100000 # broj replikacija
Tw = uniform(0, Tw_max, [N, k]) # slučajno kašnjenje
# interval na kome se generišu uniformno raspodeljeni slučajni brojevi
# je [0, Tw_max), ali to ne predstavlja problem

Q = zeros([N, k])

for i in range(N):
   D = zeros([N, k])
   for j in range(N):
      # Da li se ONU_i sudarila s nekom drugom?
      D[j, :] = (abs(Tw[j, :] - Tw[i, :]) > T)

   Q[i, :] = (sum(D, axis=0) == (N-1))
   # jedinica znači da nije bilo sudara

# verovatnoća uspeha u pojedinačnom eksperimentu
Pi = sum(Q, axis=0)/N

# usrednjavanje
P = sum(Pi)/k
U = N*P*T/(T+Tw_max)
disp(P)
disp(U)
```

12.3.4 Rezultati

Dobijeni rezultati su prikazani na slici 12.8.

Prvo što uočavamo sa grafika je da je efikasnost registracije ONU u svim slučajevima relativno mala, manja od 20%. Takođe vidimo da, za fiksan broj ONU, efikasnost isprva raste s povećavanjem trajanja slučajnog kašnjenja, odnosno prozora pronalaženja, što se objašnjava time da se smanjuje nadmetanje između ONU, a time i broj sudara. Za neku vrednost maksimalnog kašnjenja, efikasnost dostiže maksimum i potom počinje da opada, jer sada dalje povećavanje $w_{\max}$ ne doprinosi značajnijem smanjivanju broja sudara, a link ostaje neiskorišćen. S druge strane, ako je zadato maksimalno trajanje slučajnog čekanja, efikasnost registracije će isprva rasti s povećavanjem broja ONU, jer se skraćuju periodi neaktivnosti na linku; dostići će maksimum i potom početi da opada, jer će sudari postati izraženi.

Iz oblika dobijene površi zaključujemo da je moguće optimizovati proceduru registracije, tako da se (a) za zadati broj ONU odredi optimalno trajanje prozora pronalaženja, ili (b) za zadato trajanje prozora pronalaženja odredi optimalan broj ONU u mreži.



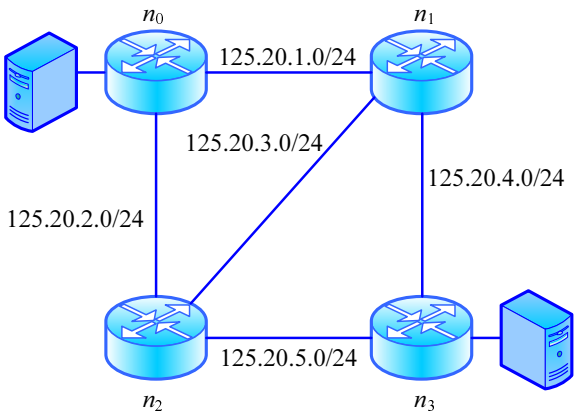
Slika 12.8: Ocena efikasnosti procedure registracije.

Pošto zavisnost U od $w_{\max}$ i N nije simetrična, ni ovi optimumi neće biti uzajamno ekvivalentni.

12.4 Dinamičko rutiranje

12.4.1 Zadatak

U mreži sa slike 12.9, izvor *on-off* UDP saobraćaja je priključen na čvor n_0, a primalac na čvor n_3. Parametri linkova su dati u tabeli 12.2.



Slika 12.9: Topologija posmatrane mreže.

Tabela 12.2: Parametri linkova.

Link	Kapacitet	Propagaciono kašnjenje	Metrika („cena”)
$n_0 - n_1$	5 Mb/s	5 ms	3
$n_0 - n_2$	5 Mb/s	2 ms	1
$n_1 - n_2$	5 Mb/s	2 ms	1
$n_1 - n_3$	5 Mb/s	2 ms	1
$n_2 - n_3$	5 Mb/s	2 ms	4

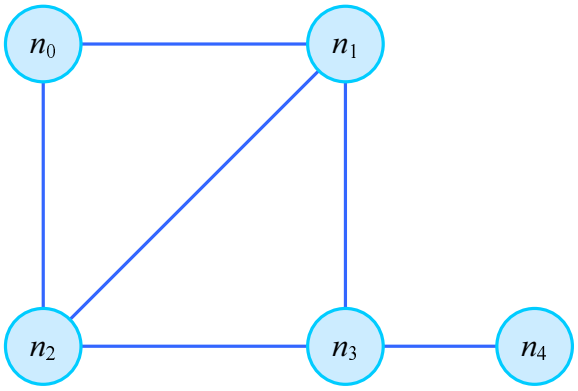
Posmatra se sledeći scenario dešavanja:

- u trenutku $t = 1$ s, izvor počinje emitovati saobraćaj vršnim protokom 448 kb/s, uz veličinu UDP paketa 210 B,
- u trenutku $t = 2$ s, ukida se link između čvorova n_1 i n_2,
- u trenutku $t = 3$ s, ukida se link između čvorova n_0 i n_1,
- u trenutku $t = 4$ s, ponovo se uspostavlja link između čvorova n_1 i n_2,
- u trenutku $t = 5$ s, isključuje se izvor.

Simulacijom je potrebno odrediti putanje saobraćaja kroz mrežu.

12.4.2 Opis simulacije

Odlučili smo se za simulaciju u programu *ns*-3. Simulacioni model je prikazan na slici 12.10.



Slika 12.10: Simulacioni model mreže.

Da bismo ilustrovali različite mogućnosti konfigurisanja simulacionog modela u *ns*-3, izvor saobraćaja smo pridružili postojećem čvoru mreže n_0, u vidu `OnOff` aplikacije, dok smo prijemnik paketa (`PacketSink`) pridružili novom čvoru n_4. Čvorovi n_3 i n_4 povezani su linkom zanemarivog kašnjenja, kome dodeljujemo IP adrese iz opsega 125.20.0.6/24.

Novo u odnosu na naš prvi primer u *ns*-3, sa strane 121 je zadavanje metrike linkova metodom `SetMetric`. Primetimo da se, iako su linkovi „tačka-tačka" dvosmerni, mogu zadati različite vrednosti metrike za dva smera prenosa.

Događaji tokom izvršavanja simulacije se specificiraju metodom `Simulator::Schedule`. U našem slučaju, njen poslednji argument se odnosi na indeks posmatranog mrežnog interfejsa, pri čemu 0 odgovara tzv. *loopback* interfejsu, dok su interfejsi ka fizičkim linkovima (u našem slučaju „tačka-tačka") numerisani po redu kojim su pravljeni. Zanimljivo je da se linkovi ne isključuju direktno, već raskidanjem veze protokolskog steka sa mrežnim interfejsom.

Izveštaje ćemo snimiti kao *trace* i `pcap` datoteke, a sačuvaćemo i tabele rutiranja posle karakterističnih događaja u mreži. Na početku programa moramo eksplicitno tražiti da se tabele rutiranja ažuriraju posle svake promene stanja linkova, tako što ćemo vrednost parametra `ns3::Ipv4GlobalRouting::RespondToInterfaceEvents` postaviti na `true`.

Listing C++ programa dat je u nastavku. Napomenimo da je prilikom njegovog izvršavanja iz komentara potrebno ukloniti dijakritičke simbole, koji su ovde zadržani radi čitljivosti teksta.

```
/* simulacija dinamičkog rutiranja
   u ns-3
 */

#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

int
main (int argc, char *argv[])
{
  // zadajemo ažuriranje tabela rutiranja posle svake izmene stanja linka:
  Config::SetDefault ("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents",
                      BooleanValue (true));
  // pravimo čvorove:
  NodeContainer c;
  c.Create (5);
  NodeContainer n0n1 = NodeContainer (c.Get (0), c.Get (1));
```

```
NodeContainer n0n2 = NodeContainer (c.Get (0), c.Get (2));
NodeContainer n1n2 = NodeContainer (c.Get (1), c.Get (2));
NodeContainer n1n3 = NodeContainer (c.Get (1), c.Get (3));
NodeContainer n2n3 = NodeContainer (c.Get (2), c.Get (3));
NodeContainer n3n4 = NodeContainer (c.Get (3), c.Get (4));

// instaliramo stek protokola na čvorovima:
InternetStackHelper internet;
internet.Install (c);

// pravimo linkove:
PointToPointHelper p2p;
p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("5ms"));
NetDeviceContainer d0d1 = p2p.Install (n0n1);

p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer d0d2 = p2p.Install (n0n2);
NetDeviceContainer d1d2 = p2p.Install (n1n2);
NetDeviceContainer d1d3 = p2p.Install (n1n3);
NetDeviceContainer d2d3 = p2p.Install (n2n3);

p2p.SetChannelAttribute ("Delay", StringValue ("0ms"));
NetDeviceContainer d3d4 = p2p.Install (n3n4);

// dodeljujemo IP adrese:
Ipv4AddressHelper ipv4;
ipv4.SetBase ("125.20.1.0", "255.255.255.0");
Ipv4InterfaceContainer i0i1 = ipv4.Assign (d0d1);

ipv4.SetBase ("125.20.2.0", "255.255.255.0");
ipv4.Assign (d0d2);

ipv4.SetBase ("125.20.3.0", "255.255.255.0");
ipv4.Assign (d1d2);

ipv4.SetBase ("125.20.4.0", "255.255.255.0");
Ipv4InterfaceContainer i1i3 = ipv4.Assign (d1d3);

ipv4.SetBase ("125.20.5.0", "255.255.255.0");
Ipv4InterfaceContainer i2i3 = ipv4.Assign (d2d3);

ipv4.SetBase ("125.20.6.0", "255.255.255.0");
Ipv4InterfaceContainer i3i4 = ipv4.Assign (d3d4);


// zadajemo metrike linkova
// (pretpostavljena vrednost je 1):
i0i1.SetMetric (0, 3);
i0i1.SetMetric (1, 3);
```

```
i2i3.SetMetric (0, 4);
i2i3.SetMetric (1, 4);

// pravimo tabele rutiranja:
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

// pravimo izvor saobraćaja
// i instaliramo ga na čvor 0:
uint16_t port = 9;   // Discard port (RFC 863)
OnOffHelper onoff ("ns3::UdpSocketFactory",
                   InetSocketAddress (i3i4.GetAddress (1), port));
onoff.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
onoff.SetAttribute ("DataRate", StringValue ("448kbps"));
onoff.SetAttribute ("PacketSize", UintegerValue (210));

ApplicationContainer apps = onoff.Install (c.Get (0));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (5.0));

// pravimo prijemnik saobraćaja
// i instaliramo ga na čvor 4:
PacketSinkHelper sink ("ns3::UdpSocketFactory",
                       Address (InetSocketAddress
                                     (Ipv4Address::GetAny (), port)));
apps = sink.Install (c.Get (4));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (5.0));

// zadajemo snimanje izveštaja:
AsciiTraceHelper ascii;
Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("primer_4.tr");
p2p.EnableAsciiAll (stream);
internet.EnableAsciiIpv4All (stream);

p2p.EnablePcapAll ("primer_4");

// postavljamo pointere:
Ptr<Node> n0 = c.Get (0);
Ptr<Ipv4> ipv4n0 = n0->GetObject<Ipv4> ();

Ptr<Node> n2 = c.Get (2);
Ptr<Ipv4> ipv4n2 = n2->GetObject<Ipv4> ();

// u t = 2 s isključujemo link n1-n2:
Simulator::Schedule (Seconds (2), &Ipv4::SetDown, ipv4n2, 2);
// u t = 3 s isključujemo link n0-n1:
Simulator::Schedule (Seconds (3), &Ipv4::SetDown, ipv4n0, 1);
// u t = 4 s uključujemo link n1-n2:
```

```
  Simulator::Schedule (Seconds (4), &Ipv4::SetUp, ipv4n2, 2);

  // snimamo tabele rutiranja:
  Ipv4GlobalRoutingHelper g;
  Ptr<OutputStreamWrapper> routingStream =
           Create<OutputStreamWrapper> ("primer_4.routes", std::ios::out);
  g.PrintRoutingTableAllAt (Seconds (1), routingStream);
  g.PrintRoutingTableAllAt (Seconds (2), routingStream);
  g.PrintRoutingTableAllAt (Seconds (3), routingStream);
  g.PrintRoutingTableAllAt (Seconds (4), routingStream);
  g.PrintRoutingTableAllAt (Seconds (5), routingStream);

  // pokrećemo simulaciju
  // i po njenom završetku uništavamo sve napravljene objekte:
  Simulator::Run ();
  Simulator::Destroy ();
}
```

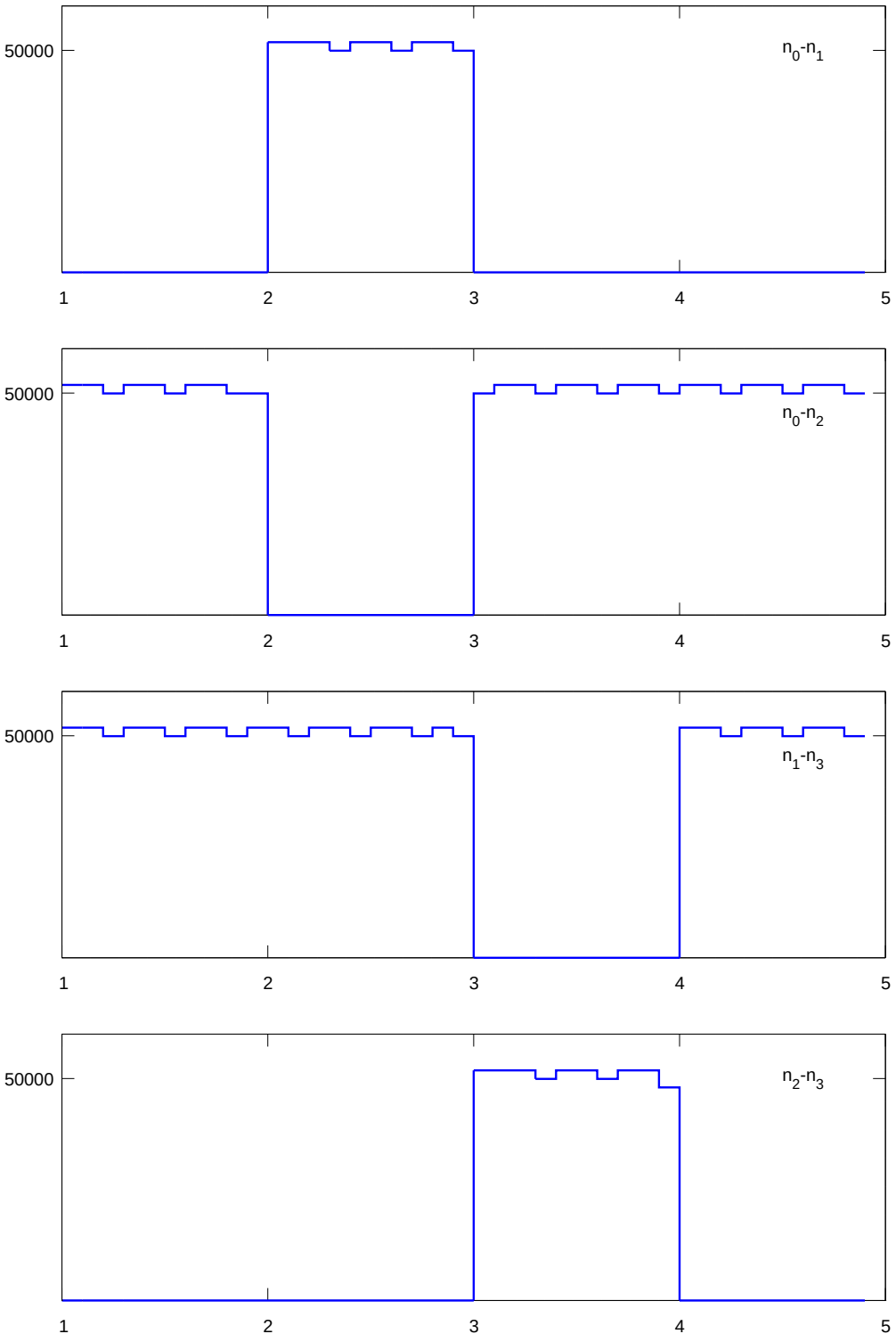
12.4.3 Rezultati

Umesto analize zapisa iz *trace* datoteke, obradili smo `pcap` datoteke u programu Wireshark, izvezli rezultate merenja saobraćaja na linkovima u `csv` datoteku i nacrtali grafike u programu GNU Octave. Rezultati su prikazani na slici 12.11.

Primenjeni protokol rutiranja (OSPF) bira rutu ka odredištu koja ima najmanju metriku; na početku je to $n_0 - n_2 - n_1 - n_3 - n_4$. Kada u trenutku $t = 2$ s bude ispao link $n_1 - n_2$, saobraćaj će se preusmeriti na rutu $n_0 - n_1 - n_3 - n_4$. U trenutku $t = 3$ s ispada i link između čvorova n_0 i n_1, pa će nova putanja biti $n_0 - n_2 - n_3 - n_4$. Kada se u trenutku $t = 4$ s bude ponovo uspostavio link između čvorova n_1 i n_2, saobraćaj će se vratiti na prvobitnu putanju $n_0 - n_2 - n_1 - n_3 - n_4$.

Gornja zapažanja potvrđuju i tabele rutranja, koje zbog obimnosti navodimo samo za čvor 1 u trenucima $t = 1$ s i $t = 2$ s.

```
Node: 0 Time: 1s Ipv4ListRouting table
  Priority: 0 Protocol: ns3::Ipv4StaticRouting
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
125.20.1.0      0.0.0.0         255.255.255.0   U     0      -      -   1
125.20.2.0      0.0.0.0         255.255.255.0   U     0      -      -   2
  Priority: -10 Protocol: ns3::Ipv4GlobalRouting
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
125.20.2.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.3.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.5.1      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.1.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
```



Slika 12.11: Protoci na linkovima.

```
125.20.3.1      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.4.1      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.4.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.5.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.6.1      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.6.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
127.0.0.0       125.20.2.2      255.0.0.0       UG    -      -      -   2
125.20.2.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.3.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.5.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
127.0.0.0       125.20.2.2      255.0.0.0       UG    -      -      -   2
125.20.1.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.3.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.4.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
127.0.0.0       125.20.2.2      255.0.0.0       UG    -      -      -   2
125.20.4.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.5.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.6.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
127.0.0.0       125.20.2.2      255.0.0.0       UG    -      -      -   2
125.20.6.0      125.20.2.2      255.255.255.0   UG    -      -      -   2

Node: 0 Time: 2s Ipv4ListRouting table
  Priority: 0 Protocol: ns3::Ipv4StaticRouting
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
125.20.1.0      0.0.0.0         255.255.255.0   U     0      -      -   1
125.20.2.0      0.0.0.0         255.255.255.0   U     0      -      -   2
  Priority: -10 Protocol: ns3::Ipv4GlobalRouting
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
125.20.2.2      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.5.1      125.20.2.2      255.255.255.255 UH    -      -      -   2
125.20.1.2      125.20.1.2      255.255.255.255 UH    -      -      -   1
125.20.4.1      125.20.1.2      255.255.255.255 UH    -      -      -   1
125.20.4.2      125.20.1.2      255.255.255.255 UH    -      -      -   1
125.20.5.2      125.20.1.2      255.255.255.255 UH    -      -      -   1
125.20.6.1      125.20.1.2      255.255.255.255 UH    -      -      -   1
125.20.6.2      125.20.1.2      255.255.255.255 UH    -      -      -   1
127.0.0.0       125.20.2.2      255.0.0.0       UG    -      -      -   2
125.20.2.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
125.20.5.0      125.20.2.2      255.255.255.0   UG    -      -      -   2
127.0.0.0       125.20.1.2      255.0.0.0       UG    -      -      -   1
125.20.1.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
125.20.3.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
125.20.4.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
127.0.0.0       125.20.1.2      255.0.0.0       UG    -      -      -   1
125.20.4.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
125.20.5.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
125.20.6.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
127.0.0.0       125.20.1.2      255.0.0.0       UG    -      -      -   1
125.20.6.0      125.20.1.2      255.255.255.0   UG    -      -      -   1
```

U trenutku $t = 1$ s, „najbolje" rute iz čvora n_0 ka svim ostalim čvorovima idu preko interfejsa 2, koji vodi ka čvoru n_2 (podsetimo se da je interfejs 0 uvek „loopback", dok su za čvor n_0 interfejsi 1 i 2 redom oni ka čvorovima n_1 i n_2, jer smo ih tim redom pravili). U trenutku $t = 2$ s, ispada link između čvorova n_1 i n_2. U tabeli rutiranja čvora n_0 stoga će rute ka čvorovima n_1 i n_3 ići preko interfejsa 1, koji vodi ka čvoru n_1.

12.5 Animacija rezultata simulacije

U ovome odeljku ćemo pokazati kako se mogu animirati rezultati simulacije u *ns*-3; za to su nam na raspolaganju alati NetAnim i pyviz. Simulacioni model je isti kao u prethodnom odeljku.

12.5.1 NetAnim

Animator NetAnim je namenjen radu s C++ programskim skriptama. Da bismo omogućili animaciju rezultata našega programa, pozvaćemo na početku modul `netanim`:

```
#include "ns3/netanim-module.h"
```

dok ćemo unutar rutine `main` dodati sledeće linije koda:

```
CommandLine cmd;
cmd.Parse (argc, argv);
```

Potom definišemo izlaznu `xml` datoteku:

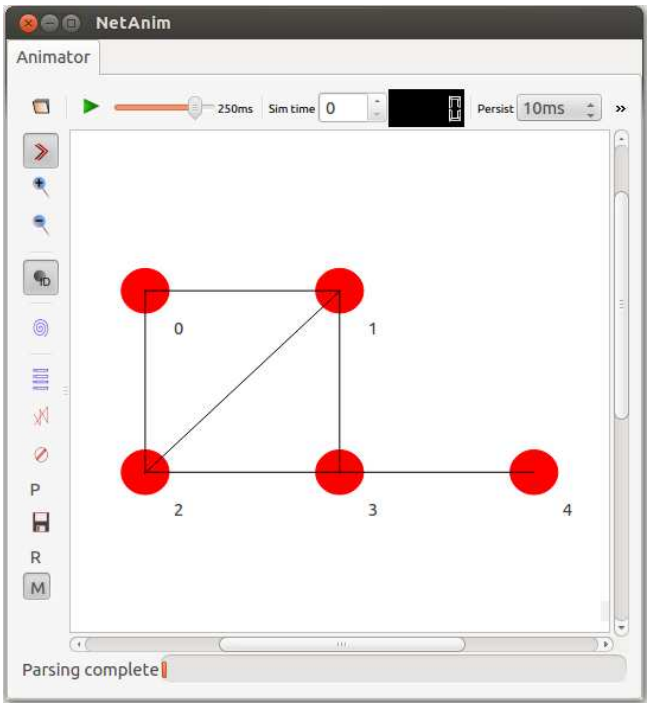
```
std::string animFile = "primer5.xml";
```

i konačno definišemo položaje čvorova. Naša mreža je fiksna, pa im dodeljujemo `ConstantPosition` model mobilnosti:

```
AnimationInterface anim (animFile);

Ptr<Node> n1 = c.Get (1);
Ptr<Node> n3 = c.Get (3);
Ptr<Node> n4 = c.Get (4);
anim.SetConstantPosition (n0, 100, 100);
anim.SetConstantPosition (n1, 200, 100);
anim.SetConstantPosition (n2, 100, 200);
anim.SetConstantPosition (n3, 200, 200);
anim.SetConstantPosition (n4, 300, 200);
```

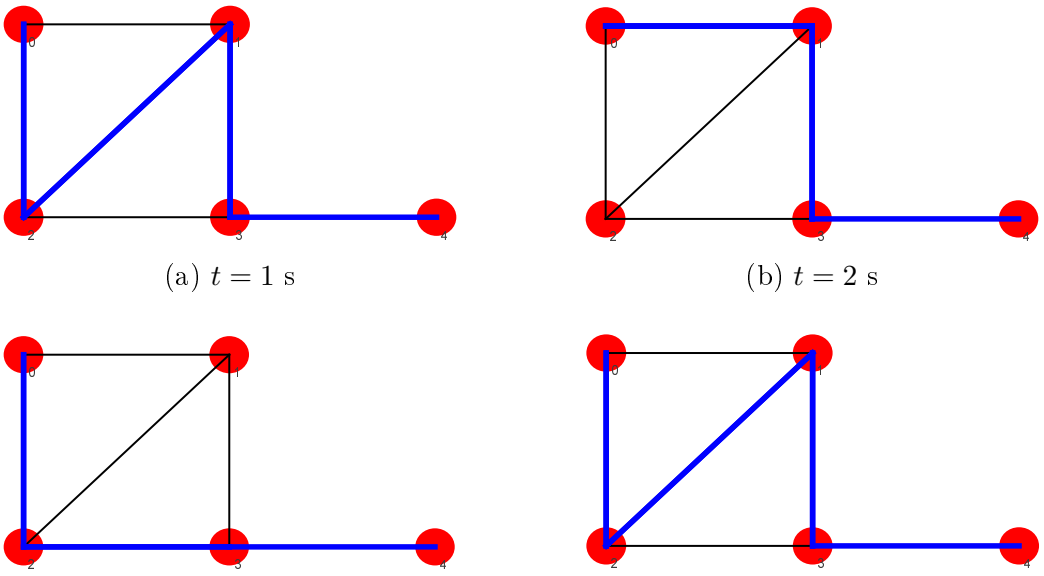
Primetimo da se koordinatni početak nalazi u gornjem levom uglu i da se koordinate čvorova zadaju preko pointera koji ukazuju na njih. Pošto smo u prethodnom odeljku napravili pointere ka čvorovima n_0 i n_2, sada ostaje da to učinimo i za čvorove n_1, n_3 i n_4.

Nakon izvršavanja simulacije, u osnovnom direktorijumu se pravi datoteka `primer5.xml`. Kada je budemo otvorili u programu NetAnim, videćemo sledeću sliku:



Slika 12.12: Prikaz topologije u animatoru NetAnim.

Pokretanjem animacije, videćemo tokove paketa po linkovima. Prikazi u karakterističnim trenucima dati su na slici 12.13.



(a) $t = 1$ s (b) $t = 2$ s

(c) $t = 3$ s (d) $t = 4$ s

Slika 12.13: Tok animacije u animatoru NetAnim.

12.5.2 Pyviz

Druga mogućnost za vizuelizaciju i animaciju rezultata simulacije koju nudi program *ns*-3 je u modulu pyviz, koji podržava obe vrste simualcionih skripata – i Python, i C++.

Da bi se omogućila animacija rezultata Python programa u pyvizu, na početku treba učitati module `visualizer` i `sys`:

```
import ns.visualizer
import sys
```

dok u telo programa treba dodati linije

```
cmd = ns.core.CommandLine()
cmd.Parse(sys.argv)
```

Program se u ovome slučaju pokreće naredbom

```
./waf --pyrun primer5.py --vis
```

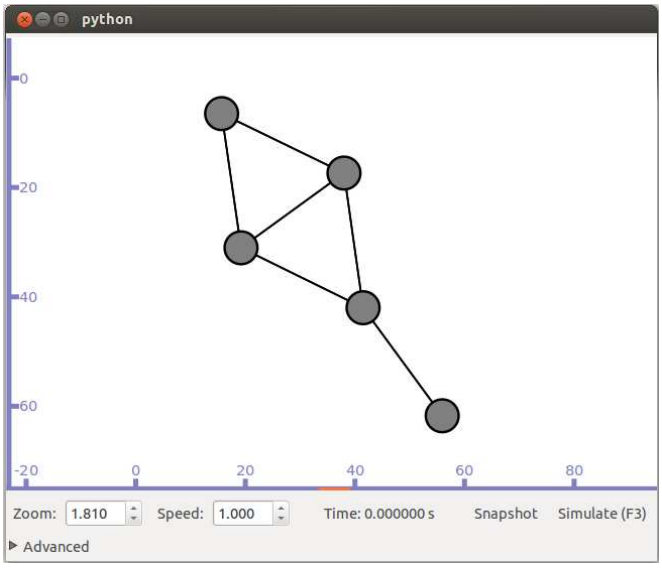
Ako se koristi C++, u glavnu programsku rutinu treba dodati linije

```
CommandLine cmd;
cmd.Parse (argc, argv);
```

Ako smo program sačuvali unutar direktorijuma `scratch`,pokrenućemo ga naredbom

```
./waf --run scratch/primer5 --vis
```

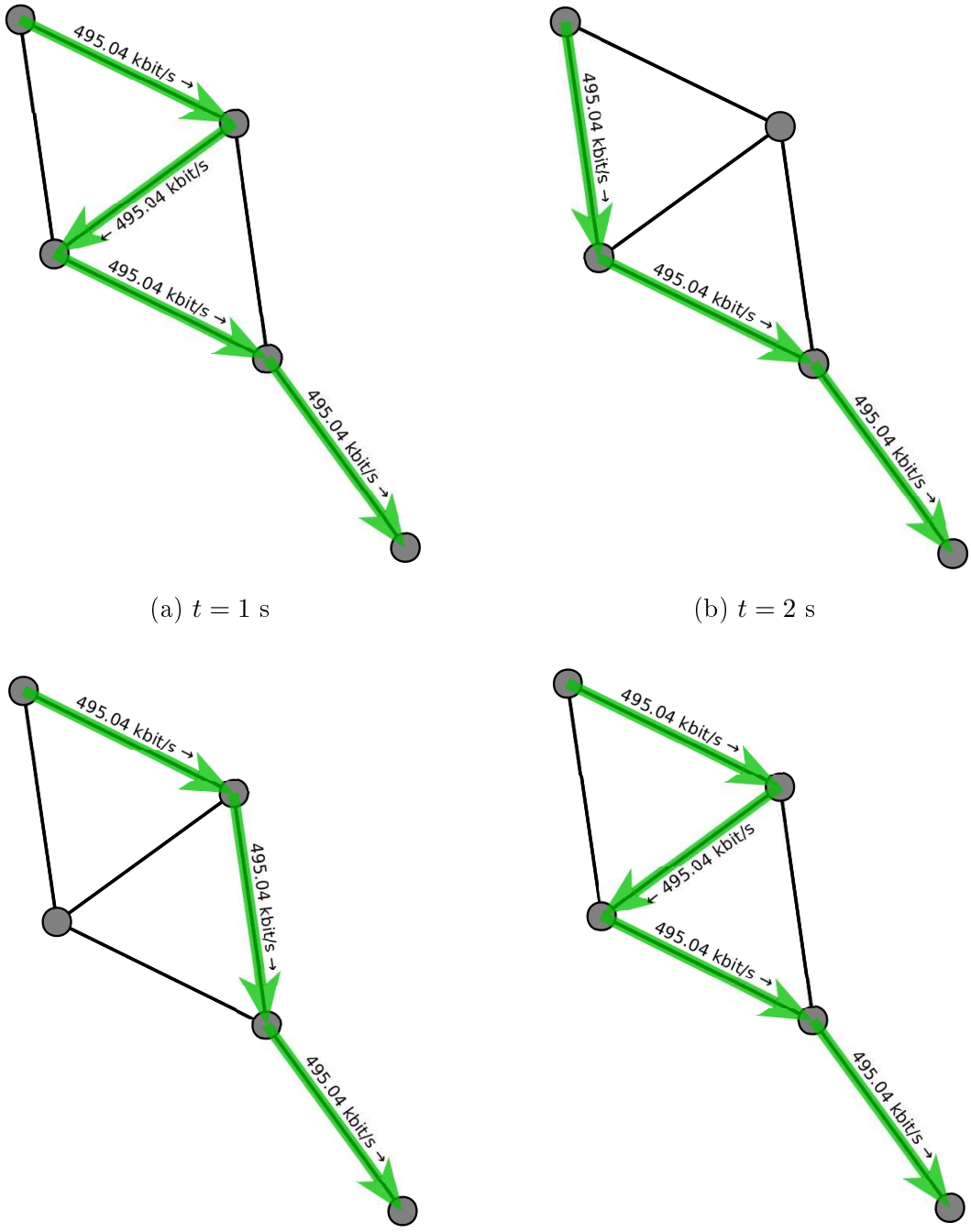
U oba slučaja se otvara prozor koji je prikazan na slici 12.14.



Slika 12.14: Animator pyviz – osnovni prikaz topologije.

Ovakvim zadavanjem animacije imamo veoma ograničenu slobodu razmeštanja čvorova unutar prozora animatora. U primeru sa slike, čvor n_0 je na vrhu, dok je prvi ispod njega čvor n_2. Tok animacije prikazan je na slici 12.15.

Primetimo da, dok animator NetAnim prikazuje pakete, pyviz prikazuje protoke na linkovima.



(a) $t = 1$ s

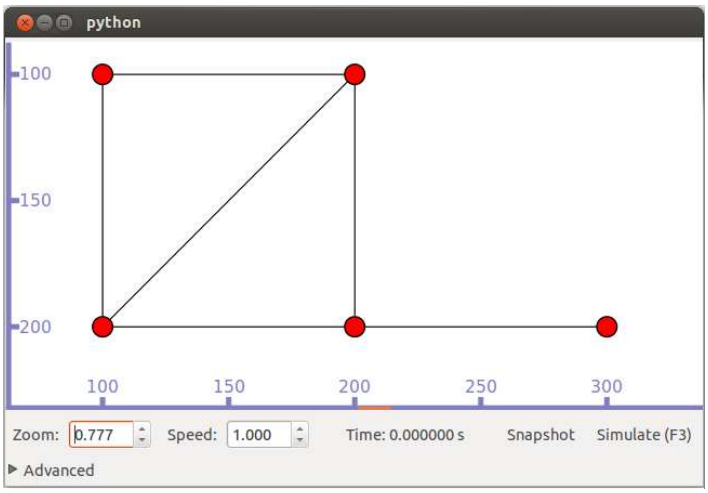
(b) $t = 2$ s

(c) $t = 3$ s

(d) $t = 4$ s

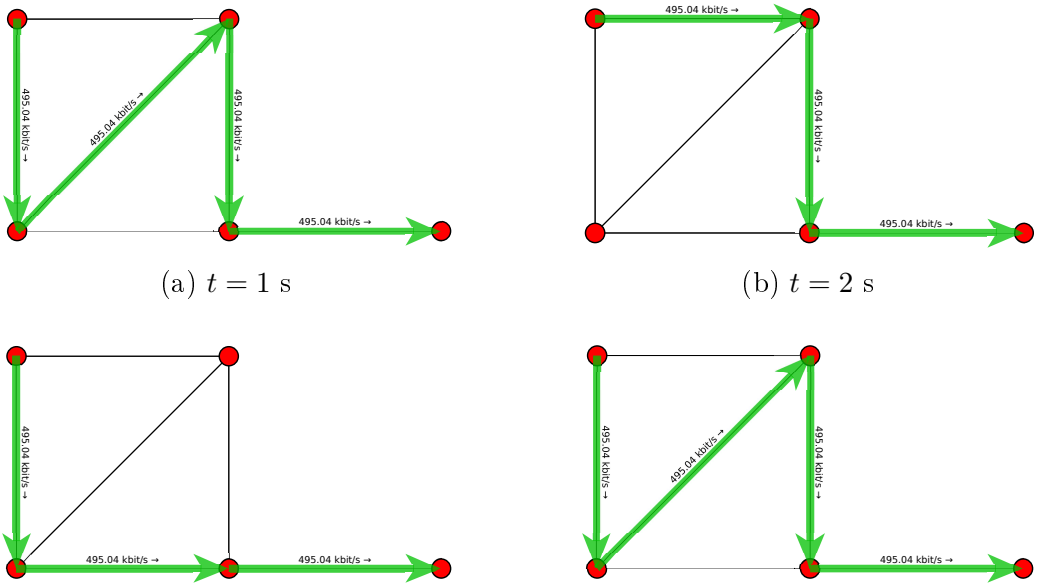
Slika 12.15: Tok animacije u animatoru pyviz za osnovni prikaz topologije.

Položaje čvorova je moguće i eksplicitno zadati, na način identičan onome kod animatora NetAnim. U tome slučaju, dobijamo prikaz kao na slici 12.16.



Slika 12.16: Animator pyviz – modifikovani prikaz topologije.

Tok animacije u ovome slučaju dat je na slici 12.17.



(a) $t = 1$ s (b) $t = 2$ s

(c) $t = 3$ s (d) $t = 4$ s

Slika 12.17: Tok animacije u animatoru pyviz za modifikovani prikaz topologije.

Literatura

[1] Averill M. Law: *Simulation Modeling & Analysis*, McGraw-Hill, 2007.

[2] Farhad Azadivar: "Simulation Optimization Methodologies", Proceedings of the 1999 Winter Simulation Conference: Simulation – a bridge to the future, Volume 1, pp. 93–100, Phoenix, USA, 1999.

[3] Jerry Banks: *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, John Wiley & Sons, 1998.

[4] Milan Bjelica, Ana Perić: "Allocation of Optimal Discovery Slots in IEEE 802.3av Networks", *AEÜ – Archiv für Elektronik und Übertragungstechnik/International Journal of Electronics and Communications*, Volume 66, Issue 3, pp. 211–213, March 2012.

[5] Milan Bjelica and Zoran Petrović: "Computer Simulation in Teaching Fundamentals of Communications: Pro et Contra", Proc. Eurocon 2003, pp. 445–448, Ljubljana, Slovenia, September 2003.

[6] A. Bruce Carlson, Paul Crilly, Paul B. Crilly: *Communication Systems*, 5th Ed, McGraw-Hill Higher Education, 2009.

[7] Christopher A. Chung: *Simulation Modeling Handbook: A Practical Approach*, CRC Press, 2004.

[8] Michael C. Fu: "Optimization for Simulation: Theory vs. Practice", *INFORMS Journal on Computing*, Vol. 14, No. 3, pp. 192–215, Summer 2002.

[9] Richard M. Fujimoto, Kalyan S. Perumalla, George F. Riley: *Network Simulation*, Morgan & Claypool Publishers, 2007.

[10] Floyd Martin Gardner, John D. Baker: *Simulation Techniques: Models of Communication Signals and Processes*, John Wiley & Sons, 1997.

[11] Fayez Gebali: *Analysis of Computer and Communication Networks*, Springer, 2008.

[12] Michel C. Jeruchim, Philip Balaban, K. Sam Shanmugan: *Simulation of Communication Systems: Modeling, Methodology, and Techniques*, Second Edition, Kluwer Academic Publishers, 2002.

[13] Jean-Yves Le Boudec: *Performance Evaluation of Computer and Communication Systems*, EPFL Press, 2011.

[14] Jesper Schmidt Hansen: *GNU Octave Beginner's Guide*, Packt Publishing, 2011.

[15] Hwei Piao Hsu: *Schaum's Outline of Theory and Problems of Probability, Random Variables, and Random Processes*, McGraw-Hill Professional, 1997.

[16] Aleksandar Karač: *Numeričke metode u inženjerstvu*, Univerzitet u Zenici, 2009.

[17] M. J. Merkle, P. M. Vasić: *Verovatnoća i statistika sa primenama i primerima*, Elektrotehnički fakultet, Beograd, 1995.

[18] Alan V. Oppenheim and George C. Verghese: *Signals, Systems, and Inference*, Class Notes for 6.011: Introduction to Communication, Control and Signal Processing, Massachusetts Institute of Technology, Spring 2010.

[19] Sigurdur Ólafsson, Jumi Kim: "Simulation Optimization", Proceedings of the 2002 Winter Simulation Conference: Exploring new frontiers, pp. 79–84, San Diego, USA, 2002.

[20] Ana Perić, Milan Bjelica: "Virtual Laboratories in Teaching Computer Networks", in *Computer Simulations: Technology, Industrial Applications and Effects on Learning*, Nova Publishers, 2012.

[21] Gerardo Rubino and Bruno Tuffin (Eds.): *Rare Event Simulation using Monte Carlo Methods*, John Wiley & Sons Ltd, 2009.

[22] *ns-3 Tutorial*, Release ns-3-dev, ns-3 project, December 2011.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

621.39:004.942(075.8)(0.034.2)

БЈЕЛИЦА, Милан, 1977-
Modeliranje i simulacija u telekomunikacijama [Elektronski izvor] / Milan Bjelica. - Beograd : Elektrotehnički fakultet, 2013

Sistemski zahtevi: Nisu navedeni. - Način dostupa (URL): http://www.etf.bg.ac.rs/etf_files/udzbenici/Modeliranje_i_simulacija_u_telekomunikacijama. - Nasl. sa naslovne strane dokumenta. - Opis izvora dana 05. 02. 2013. - Na vrhu nasl. ekrana: Udžbenik Elektrotehničkog fakulteta u Beogradu. - Sadrži bibliografiju.

ISBN 978-86-7225-053-4

a) Телекомуникациона мерења - Моделирање
b) Телекомуникациона мерења - Симулација
COBISS.SR-ID 196580364