Teoria

- Johdanto simulointiin
- Simuloinnin kulku -- prosessin realisaatioiden tuottaminen
- Satunnaismuuttujan arvonta annetusta jakaumasta
  - Johdanto ja pseudosatunnaislukujen generointi
  - Eri menetelmiä satunnaismuuttujien generointiin annetusta jakaumasta
- Tulosten keruu ja analyysi
- Varianssinreduktiotekniikoista

Satunnaismuuttujan arvonta annetusta jakaumasta

Satunnaismuuttujien generointi on liian tärkeä asia jätettäväksi sattuman varaan !

- *Hyvä statistiikka* (noudattaa jakaumaa) ja riippumattomuus
  - sulkee pois "hatusta vetämisen"
- *Toistettava sekvenssi*
  - sulkee pois todellisen arvonnan tai fysikaalisten prosessien (radioaktiivisuus) käytön
- *Pitkä sarja* eri lukuja
  - sulkee pois ennalta tehtyjen taulukoiden käytön
- *Nopea generointi*
  - sulkee pois π:n tai *e*:n tai vastaavan desimaalien käytön

Satunnaismuuttujan arvonta annetusta jakaumasta

- Pohjana ns. **(pseudo)satunnaislukujen generointi**
  - tavoitteena on tuottaa riippumattomia U(0,1)-jakautuneita (tasanjak) satunnaismuuttujia
- Haluttuun jakaumaan päästään U(0,1)-jakaumasta esimerkiksi jollakin seuraavista menetelmistä:
  - **diskretointi** (=> Bernoulli(p), Bin(n,p), Poisson(a), Geom(p))
  - **uudelleen skaalaamalla** (=> U(a,b))
  - **kertymäfunktion käännös** (=> Exp(λ))
  - **muut muunnokset** (=> N(0,1) => N(μ,σ^2))
  - **hylkäysmenetelmällä** (mikä tahansa jakauma)
  - **jakauman karakterisoinnilla** (palautus muihin jakaumiin) (=> Erlang(n,λ), Bin(n,p))
  - **yhdistelmämenetelmällä**
- Simulointia tukevista kielistä, kirjastoista ja työkaluista löytyy yleensä valmiit rutiinit kaikkia tavallisimpia jakaumia noudattavien satunnaislukujen generointiin

Satunnaislukujen generointi

- Satunnaislukugeneraattorilla tarkoitetaan algoritmia, joka tuottaa sarjan (näennäisesti) satunnaisia kokonaislukuja Z_i jollakin välillä 0,1,…,m-1
  - tuotettu sarja on aina jaksollinen (tavoitteena mahdollisimman pitkä jakso)
  - generoidut luvut eivät “tiukasti ottaen” ole ollenkaan satunnaisia vaan deterministisiä (siis pseudosatunnaisia)
  - käytännössä homma kuitenkin toimii, kunhan luvut generoidaan “huolella”
- Satunnaislukugeneraattorin generoimien satunnaislukujen “satunnaisuus“ on testattava tilastollisin testein
  - saadun empiirisen jakauman tasaisuus joukossa {0,1,…,m-1}
  - generoitujen satunnaislukujen välinen riippumattomuus (käytännössä korreloimattomuus)
- Yksinkertaisimpia ovat ns. *lineaariset kongruentiaaliset generaattorit* (linear congruential generator). Näistä erikoistapauksena saadaan ns. *multiplikatiiviset kongruentiaaliset generaattorit* (multiplicative congruential generator).
- Kummassakin tapauksessa uusi satunnaisluku määräytyy algoritmisesti välittömästi edellisestä, ts. $Z_{i+1} = f(Z_i)$.
- Muita menetelmiä: additive congruential generators, shuffling

Linear congruential generator (LCG)

- Lineaarinen kongruentiaalinen satunnaislukugeneraattori tuottaa satunnaisia kokonaislukuja Z_i joukosta {0,1,…,m-1} seuraavalla kaavalla (jakso korkeintaan m):

$$Z_{i+1} = (aZ_i + c) \bmod m$$

- Tällainen generaattori siis määritellään antamalla parametrit a, c ja m
- Lisäksi tarvitaan ns. siemenluku Z_0
- Parametrit on valittava huolella; muutoin tuloksena kaikkea muuta kuin satunnaisia lukuja
- Tietyin edellytyksin jaksoksi saadaan maksimiarvo m
  - esim. m muotoa 2^b, c pariton, a muotoa 4*k +1

Multiplicative congruential generator (MCG)

- Multiplikatiivinen kongruentiaalinen satunnaislukugeneraattori tuottaa satunnaisia kokonaislukuja Z_i joukosta {0,1,…,m-1} seuraavalla kaavalla:

$$Z_{i+1} = (aZ_i) \bmod m$$

- Kyseessä on siis LCG:n erikoistapaus valinnalla c = 0
- Tällainen generaattori määritellään antamalla parametrit a ja m
- Lisäksi tarvitaan siemenluku Z_0
- Parametrit on tässäkin tapauksessa valittava huolella; muutoin tuloksena kaikkea muuta kuin satunnaisia lukuja
- Valinnalla m = 2^b (millä tahansa b) jaksoksi saadaan korkeintaan 2^(b-2)
- Parhaimmillaan jaksoksi saadaan kuitenkin m - 1
  - esim. m alkuluku (prime) ja a valitaan sopivasti

U(0,1)-jakautuneen sm:n generointi (tasainen jakauma välillä (0,1))

- Jos (pseudo)satunnaislukugeneraattori tuottaa (pseudo)satunnaisen kokonaisluvun I joukosta {0,1,…,M-1}, niin normeerattu muuttuja U = I/M noudattaa likimain U(0,1)-jakaumaa (tasainen jakauma välillä (0,1))

Siemenlukujen käyttö

(riippuen rand-algoritmista jotkin ohjeista ovat relevantteja / irrelevantteja)

- Älä käytä arvoa 0
- Vältä parillisia arvoja
- Älä käytä yhtä satunnaislukusekvenssiä useaan tarkoitukseen
  - jos lukuja (u_1, u_2, u_3, …) on käytetty saapumisväliaikojen generointiin, samaa sekvenssiä ei pidä käyttää palveluaikojen generointiin
- Eri sekvenssejä käytettäessä varmistu, että ne eivät ole miltään osin päällekkäisiä
  - jos jonkin sekvenssin siemenluku sisältyy toiseen sekvenssiin, niin siitä eteenpäin sekvenssit ovat identtisiä
  - jos pitää generoida 10000 väliaikaa ja 10000 palveluaikaa, voidaan väliaikojen generointiin käytettävät satunnaisluvut laskea siemenluvusta u_0 alkaen (u_1, u_2,...,u_{10000})
  - palveluaikojen generointiin voidaan käyttää lukuja (u_{10001},…,u_{2000}) eli siemenluku on u_{10000}
  - jos satunnaislukusekvenssejä ei talleteta, vaan lukuja generoidaan tarpeen mukaan, on toisen sekvenssin siemenluku u_{10000} selvitettävä etukäteen laskemalla kaikki luvut (u_1, u_2,...,u_{10000}); nämä lasketaan sitten uudelleen simuloinnin aikana ensimmäistä sekvenssiä tuotettaessa

Teoria

- Johdanto simulointiin
- Simuloinnin kulku -- prosessin realisaatioiden tuottaminen
- Satunnaismuuttujan arvonta annetusta jakaumasta
  - Johdanto ja pseudosatunnaislukujen generointi
  - Eri menetelmiä satunnaismuuttujien generointiin annetusta jakaumasta
- Tulosten keruu ja analyysi
- Varianssinreduktiotekniikoista

S-38.3148 Tietoverkkojen simulointi / Satunnaismuuttujien generointi

Diskreetin sm:n generointi

- Olk. U ~ U(0,1)
- Olk. X diskreetti sm arvojoukolla S = {0,1,2,…,n} tai S = {0,1,2,...}
- Merk. F(i) = P{X ≤ i}
- Tällöin sm Y, missä

$$Y = \min\{i \in S \,|\, F(i) \geq U\}$$

noudattaa samaa jakaumaa kuin X (Y ~ X).

- Tätä kutsutaan diskretointimenetelmäksi. Itse asiassa kyseessä on diskreetin jakauman kertymäfunktion käännös
- Esim. Bernoulli(p)-jakauma:

$$Y = 1_{\{U > 1-p\}} = \begin{cases} 0, & U \leq 1-p \\ 1, & U > 1-p \end{cases}$$

Diskreetin sm:n generointi (jatkoa)

- Edellä kuvattu menetelmä, jossa satunnaisluvun U ~ U(0,1) arvoa verrataan peräkkäin kertymäfunktion arvoihin on täysin yleinen
- Monen vertailun tekeminen on kuitenkin laskennallisesti hidasta (sm:ien generointi on simulaattorin ydinsilmukassa ja pitää tapahtua hyvin nopeasti)
- Joissakin yksinkertaisissa tapauksissa vertailuihin perustuva generointi voidaan korvata menetelmällä, jossa suoraan satunnaisluvusta U lähtien halutun diskreetin sm:n arvo voidaan laskea yksinkertaisella kaavalla
- Eräitä esimerkkejä on seuraavassa taulukossa

Generointi yksinkertaisista diskreeteistä jakaumista

- Seuraavassa $U,\ U_1, \ldots,\ U_n$ tarkoittavat riippumattomia U(0,1)-jakautuneita sm:jia
- $\text{int}(X) = \lfloor X \rfloor = X$:n kokonaisosa

Jakauma	Generointikaava
Symm. kaksiarvoinen {0,1} jakauma $\mathrm{P}\{X=0\} = \mathrm{P}\{X=1\} = 0.5$	$\text{int}(2U)$ tai $\text{int}(U+0.5)$
Kaksiarvoinen {0,1} jakauma $\mathrm{P}\{X=0\} = 1-p,\ \mathrm{P}\{X=1\} = p$	$\text{int}(U+p)$
Kaksiarvoinen {-1,1} jakauma $\mathrm{P}\{X=0\} = 1-p,\ \mathrm{P}\{X=1\} = p$	$2\ \text{int}(U+p) - 1$
Kolmiarvoinen {0,1,2} jakauma tn:t $= \{1-p_1-p_2,\ p_1,\ p_2\}$	$\text{int}(U+p_2) + \text{int}(U+p_1+p_2)$
Tasainen diskreetti jakauma $\{0,\ 1,\ 2, \ldots, n-1\}$	$\text{int}(n\,U)$
Tasainen diskreetti jakauma $\{1,\ 2,\ 3, \ldots, n\}$	$\text{int}(n\,U) + 1$
Binomijakauma $\mathrm{Bin}(n,p)$	$\sum_{i=1}^{n} \text{int}(U_i + p)$

Diskreetin sm:n generointi (jatkoa …)

- Usein tilanne on kuitenkin, että diskreetin sm:n generointi suoraan annetusta muuttujasta U~U(0,1) ei ole mahdollista
- Tällöin X:n diskreetti kertymäfunktio F(x) täytyy laskea ja tallettaa johonkin tietorakenteeseen
- Yksinkertaiset haut:
  – talletetaan F(x):n arvot vektoriin ja haetaan alkaen alusta (lineaarinen haku)
  – parannettu haku: mikä tahansa tehokkaampi haku, esim. binäärihaku
  – parannettu binäärihaku: otetaan talteen se indeksin arvo, jossa F(x) on likipitäen 0.5, jolloin saadaan yhdellä vertailulla selville ”kummalla puolella” haun tulos on, ja käytetään sitten binäärihakua (tai lineaarista hakua)
- Optimaalinen haku (pienin määrä vertailuja):
  – esitetään F(x) Huffmanin puun muodossa

S-38.3148 Tietoverkkojen simulointi / Satunnaismuuttujien generointi

Huffmanin puun käyttö diskreetin sm:n generointiin

- Talletetaan jakauma Huffmanin puu tietorakenteeseen, jolloin todennäköisimpiin vaihtoehtoihin on lyhin reitti
- Huffmanin puu luodaan seuraavasti
  – Luo solmujen joukko, joissa on tieto jakauman arvoista ja kunkin arvon todennäköisyydestä. Järjestä solmut todennäköisyyksien mukaan.
  – Ota listan kaksi ensimmäistä solmua (kaksi pienintä todennäköisyyttä) ja luo niille isäntäsolmu siten että pienempi jälkeläisistä on vasemmalla puolella. Uuden solmun todennäköisyys on nyt kahden jälkeläisen todennäköisyyksien summa. Uusi solmu laitetaan listaan vastaavaan paikkaan.
  – Toista edellistä kunnes listassa on vain yksi elementti, joka on puun juuri.
- Algoritmi näytteiden generointiin
  – Arvo sm U ~ U(0,1) ja vertaa sitä vasemmalla olevan jälkeläisen todennäköisyyteen
  – Jos arvo on pienempi, siirry vasempaan haaraan. Jos arvo on suurempi, vähennä vasemman jälkeläisen todennäköisyys sm U:sta ja siirry oikeaan haaraan. Vertaa todennäköisyyttä seuraavaan vasemmanpuoleiseen jälkeläiseen.
  – Toista edellistä kunnes on saavuttu lehtisolmuun.

Esimerkki Huffmanin puusta

```
Iteration 1
-----------
List: (0.1, 1), (0.15, 4), (0.20, 3), (0.25, 5), (0.30, 2)
Tree:
         0.25:*
         /    \
    0.10:1   0.15:4

Iteration 2
-----------
List: (0.20, 3), (0.25,*), (0.25, 5), (0.30, 2)
Tree:
         0.45:*
         /   \
    0.20:3  0.25:*
             /    \
         0.10:1  0.15:4

Iteration 3
-----------
List: (0.25, 5), (0.30, 2), (0.45,*)
Tree:
          0.55:*
         /     \
     0.25:5  0.30:2
```

value	f(x)	F(x)
1	0.10	0.10
2	0.30	0.40
3	0.20	0.60
4	0.15	0.75
5	0.25	1.00

```
Iteration 4
-----------
List: (0.45,*), (0.55,*)
Tree:
              _____1.0:*______
             /                \
         0.45:*              0.55:*
         /   \               /    \
    0.20:3  0.25:*       0.25:5  0.30:2
            /    \
        0.10:1  0.15:4
```


Tasajakautuneen sm:n generointi

- Olk. $U \sim U(0,1)$
- Tällöin $X = a + (b - a)U \sim U(a,b)$
- Mielivaltaiseen tasajakaumaan päästään siis skaalauksella

Kertymäfunktion käännös -menetelmä

- Olk. U ~ U(0,1)
- Olk. X jatkuva sm arvojoukolla S = [0,∞)
- Oletetaan, että X:n kf F(x) = P{X ≤ x} on aidosti kasvava, jolloin sillä on käänteisfunktio $F^{-1}(y)$
- Tällöin sm Y, missä

$$Y = F^{-1}(U)$$

noudattaa samaa jakaumaa kuin X (Y ~ X).

- Tätä kutsutaan kertymäfunktion käännös -menetelmäksi (inverse transformation)
- Todistus: Koska P{U ≤ z} = z kaikilla z välillä [0,1], niin pätee

$$P\{Y \le x\} = P\{F^{-1}(U) \le x\} = P\{U \le F(x)\} = F(x)$$

Näin ollen Y ~ X.

S-38.3148 Tietoverkkojen simulointi / Satunnaismuuttujien generointi

Eksponentiaalisen sm:n generointi

- Olkoon U ~ U(0,1)
- Olkoon X ~ Exp(λ)
- X:n kf F(x) = P{X ≤ x} = 1 - exp(-λx) on aidosti kasvava, joten sillä on käänteisfunktio $F^{-1}(y) = -(1/\lambda)\ln(1 - y)$
- Koska U ~ U(0,1), myös 1 - U ~ U(0,1)
- Näin ollen (kertymäfunktion käännös -menetelmän mukaan) saadaan
- <u>Algoritmi</u>

$$X = F^{-1}(1-U) = -(1/\lambda)\log U$$

Weibullin jakaumaa noudattavan sm:n generointi

- Weibullin jakauma W(λ,β) on yleistys eksponenttijakaumasta
- Sm:n X ~ W(λ,β) kf on F(x) = P{X ≤ x} = 1 - exp(-(λx)$^\beta$)
- Tämän käänteisfunktio on $F^{-1}(y) = (1/\lambda)[-\ln(1 - y)]^{1/\beta}$
- Algoritmi

$$\boxed{X = F^{-1}(1-U) = (1/\lambda)(-\log U)^{1/\beta}}$$

Poisson-prosessin mukaisten saapumisten generointi

- Tärkein tietoliikennejärjestelmien analyysissä käytetty saapumisprosessin malli on **Poisson-prosessi**
- Poisson-prosessia, jonka saapumisintensiteetti on λ, voidaan tunnetusti karakterisoida prosessina, jossa saapumisväliajat ovat riippumattomia Exp(λ)-jakautuneita sm:ia
- Merkitään asiakkaan n saapumishetkeä t_n:llä
- Saapumishetket voidaan peräkkäin generoida kaavalla $t_{n+1} = t_n + X_n$ missä X_n ~ Exp(λ)

- Toinen tapa generoida Poisson-prosessin realisaatio välille (0,T) on:
  - arvo saapumisten kokonaislukumäärä N Poisson-jakaumasta N ~ Poisson(λT)
  - arvo jokaisen pisteen paikka t_n välillä (0,T) tasaisesta jakaumasta t_n ~ U(0,T)
  - pisteet $t_1, t_2, \ldots, t_n$ suuruusjärjestykseen asetettuna muodostavat realisaation Poisson-prosessin saapumishetkille

Epästationäärisen Poisson-prosessin mukaisten saapumisten generointi (1)

- Toisinaan on tilanteita, joissa oletus Poisson saapumisintensiteetin vakioisuudesta ei päde (vrt. aikarajoitetut äänestykset/kilpailut)
  - Poisson-prosessia, jonka saapumisintensiteetti on ajan funtio $\lambda(t)$, kutsutaan epästationääriseksi Poisson prosessiksi
- Kaksi tapaa: ohennusmenetelmä tai käännösmenetelmä
- Ohennusmenetelmä
  - Poisson prosessi intensiteetillä λ, jossa saapuminen hyväksytään todennäköisyydellä p (ei hyväksytä tn:llä 1-p), on edelleen Poisson prosessi, mutta intensiteetillä $p\lambda$
  - Simuloinnissa käytettävä $\lambda(t)$ on tunnettu ja sille voidaan laskea yläraja $\lambda^* \geq \lambda(t)$, $\forall t$
  - Idea: Generoidaan saapumisia intensiteetilla λ^* ja hyväksytään saapuminen hetkellä t' todennäköisyydellä $\lambda(t')/\lambda^*$
- Algoritmi: olkoon asiakkaan n saapumishetki t_n
  1. Aseta $\tau = t_n$
  2. Generoi U_1, $U_2 \sim U(0,1)$
  3. Aseta $\tau = \tau - (1/\lambda^*) \ln U_1$
  4. Jos $U_2 \leq \lambda(\tau)/\lambda^*$ palauta $t_{n+1} = \tau$, muuten palaa kohtaan 2
- Ongelma: ”Ylimääräisten” saapumistapahtumien generointi
  - tehoton, jos λ^* on suuri verrattuna $\lambda(t)$:n yleiseen arvoon

Epästationäärisen Poisson-prosessin mukaisten saapumisten generointi (2)

- Kääntämismenetelmä: vastaa kertymäfunktion kääntämistä
- Määritellään funktio $\Lambda(t)$

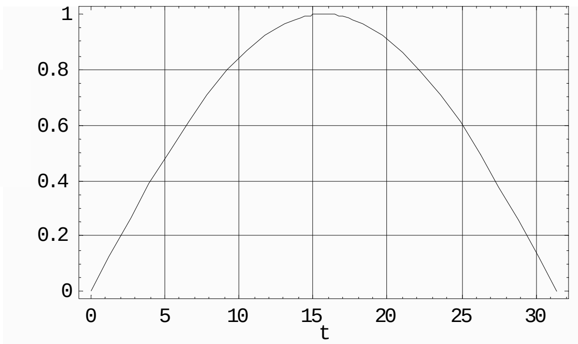
$$\boxed{\Lambda(t) = \int_0^t \lambda(y)\, dy}$$

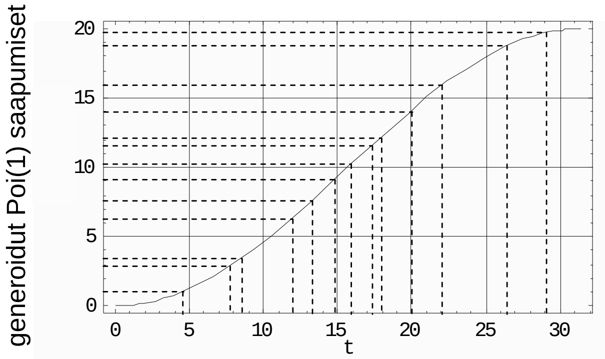
  - $\Lambda(t)$ on välillä [0, t] saapuneiden asiakkaiden keskimääräinen lukumäärä
- Idea: Generoidaan ensin saapumishetkiä τ_n Poisson intensiteetillä 1 ja suoritetaan muunnos $t_n = \Lambda^{-1}(\tau_n)$, missä $\Lambda^{-1}(y)$ on $\Lambda(y)$:n käänteisfunktio. Tällöin saapumishetket t_n noudattavat Poisson prosessia intensiteetillä $\lambda(t)$
- Algoritmi: olkoon τ_n n:nen asiakkaan saapumishetki Poisson intensiteetillä 1, olkoon t_n todellinen saapumisaika intensiteetillä $\lambda(t)$
  1. Generoi $U_1 \sim U(0,1)$
  2. Aseta $\tau_{n+1} = \tau_n - \ln U_1$
  3. Palauta $t_{n+1} = \Lambda^{-1}(\tau_{n+1})$
- Etu: kaikki saapumistapahtumat saadaan käytettyä
  - Ongelmana, että käänteisfunktion ratkaiseminen voi olla vaikeaa/mahdotonta

Epästationäärisen Poisson-prosessin mukaisten saapumisten generointi (3)

- Esimerkki kääntämismenetelmästä:

$$\lambda(t) = \sin(t/10) \quad , \quad t \in [0, 10\pi] \quad \Leftrightarrow \quad \Lambda(t) = 10(1 - \cos(t/10))$$





toteutuneet Poi(λ(t)) saapumiset

S-38.3148 Tietoverkkojen simulointi / Satunnaismuuttujien generointi

Normaalijakautuneen sm:n generointi

- Olk. U ja V riippumattomia U(0,1)-jakautuneita sm:ia
- Ns. Box-Müller -menetelmän mukaan sm:t X ja Y, missä

$$X = \sqrt{-2\ln U}\cos(2\pi V)$$
$$Y = \sqrt{-2\ln U}\sin(2\pi V)$$

ovat riippumattomia N(0,1)-jakautuneita sm:ia

- Mielivaltaiseen normaalijakaumaan päästään skaalaamalla: $\mu + \sigma X \sim N(\mu,\sigma^2)$