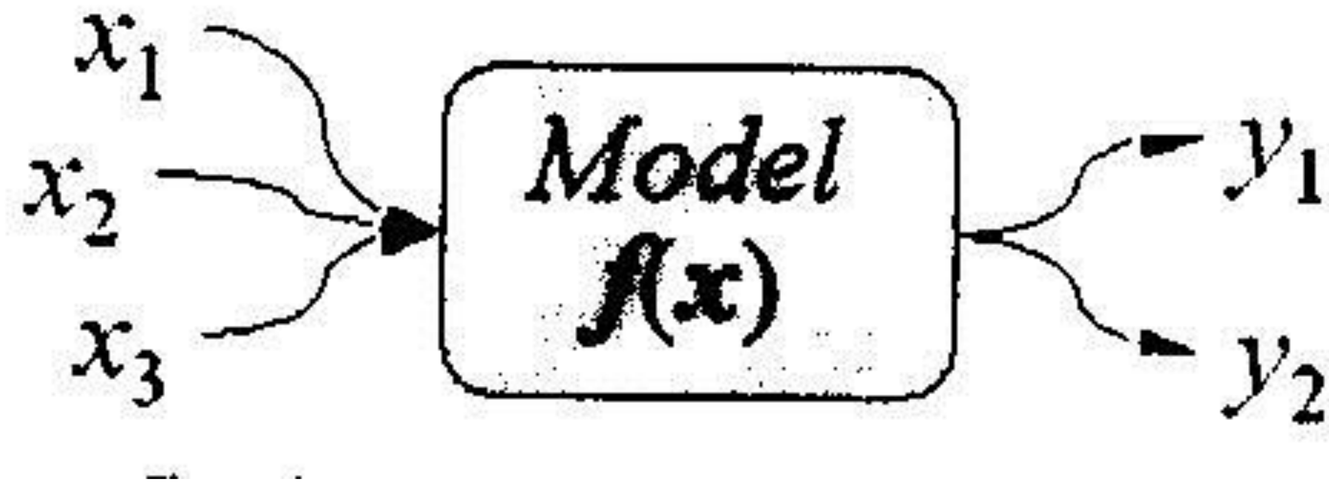
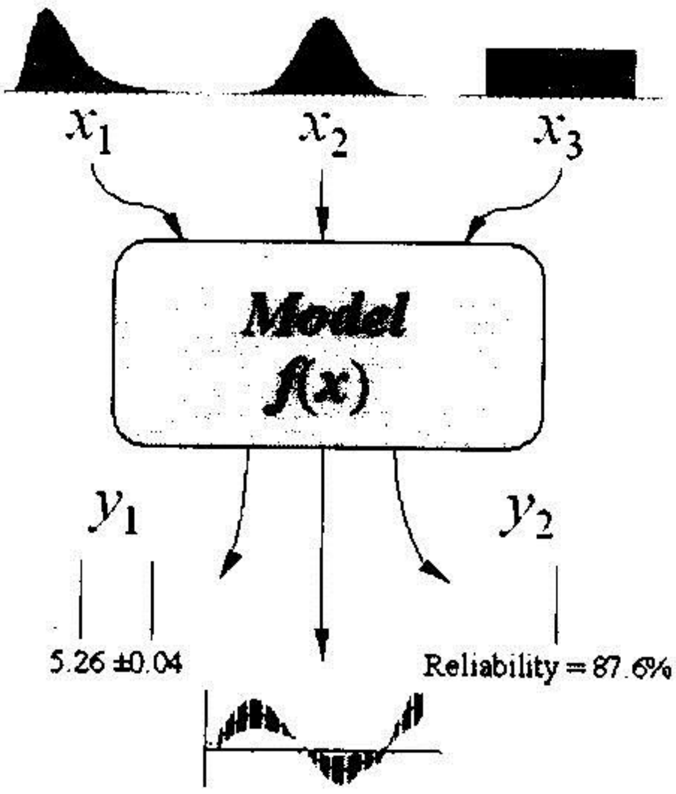
Véletlen számok generálása

Monte Carlo módszer

Determinikus modell



Stochastikus modell



Monte Carlo módszer (MCM)

1. Az MCM a determinisztikus modellek iteratív értékelésére szolgáló módszer, amely bemeneti paraméterként véletlen számokat használ.

2. A sztochasztikus numerikus szimulációs módszer egy módszer, melynek célja a rendszer utánzása.

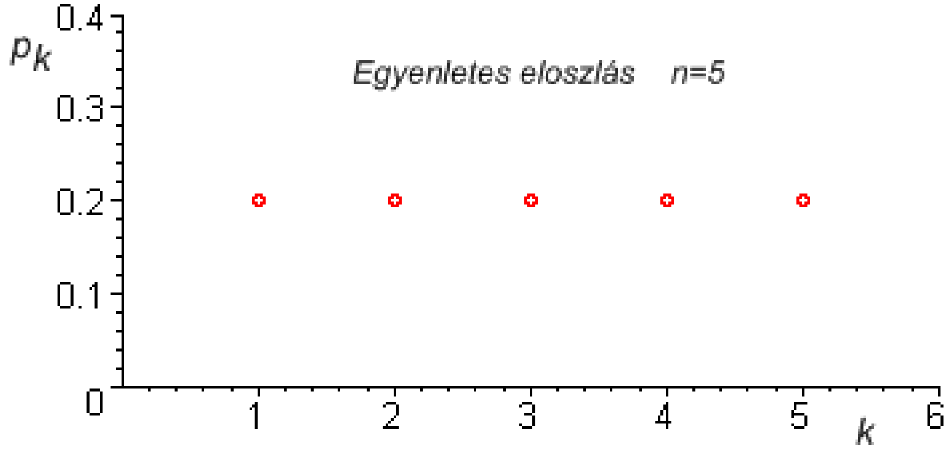
MCM lépések

$$Y = F(X)$$
$$Y = (y_1, \ldots y_m)$$
$$X = (x_1, \ldots x_n)$$

1. Modell létrehozása
2. Az X_i véletlen szám generálása
3. A modell kimenetének a meghatározása – kimenet Y_i
4. Az 2,3 pre i=1, ...,N lépések ismétlése
5. Az eredmények kiértékelése

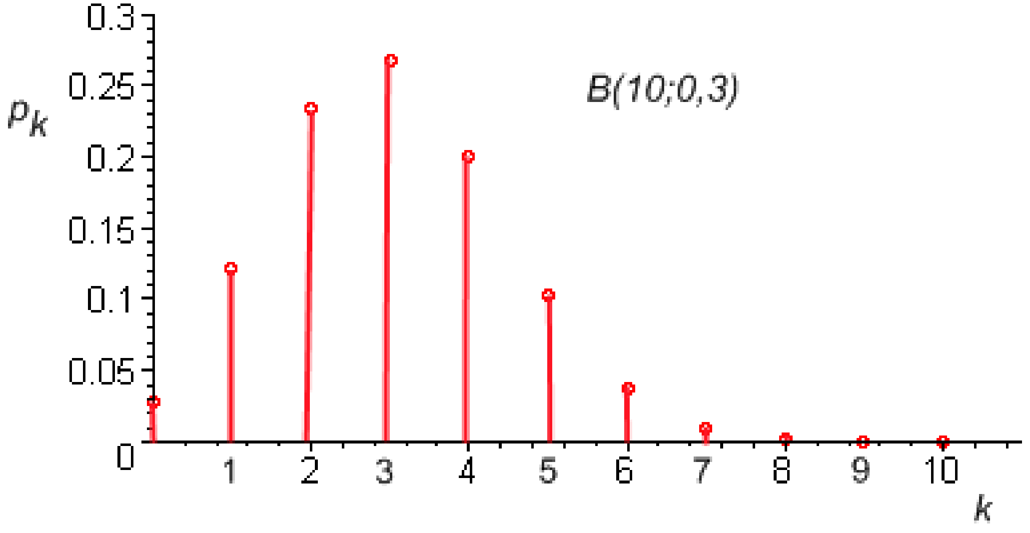
Diszkrét egyenletes eloszlás

$$p_k = P(X = x_k) = \frac{1}{n}, 1 \leq k \leq n$$



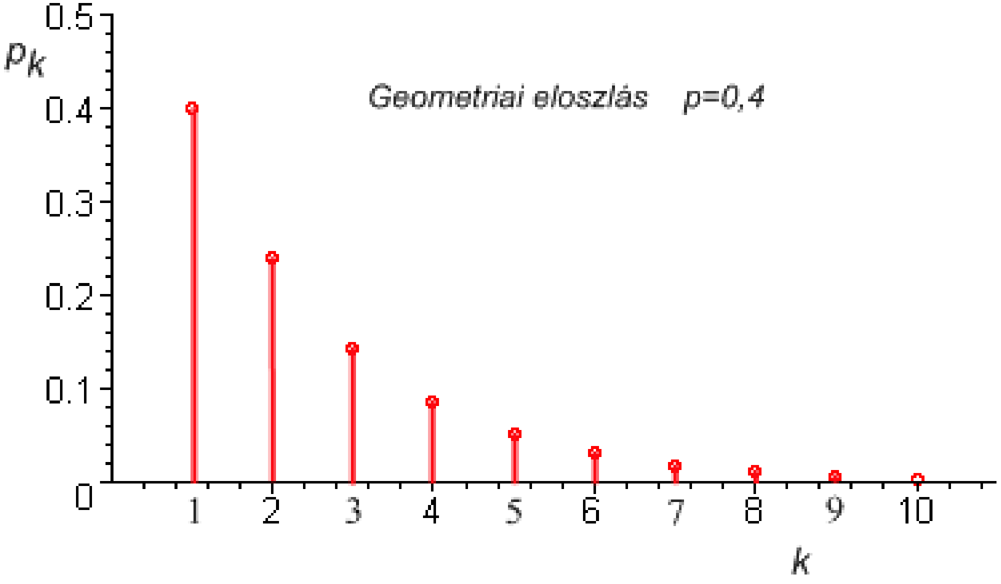
BINOMIÁLIS ELOSZLÁS

$$p_k = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, k = 0, \ldots n$$



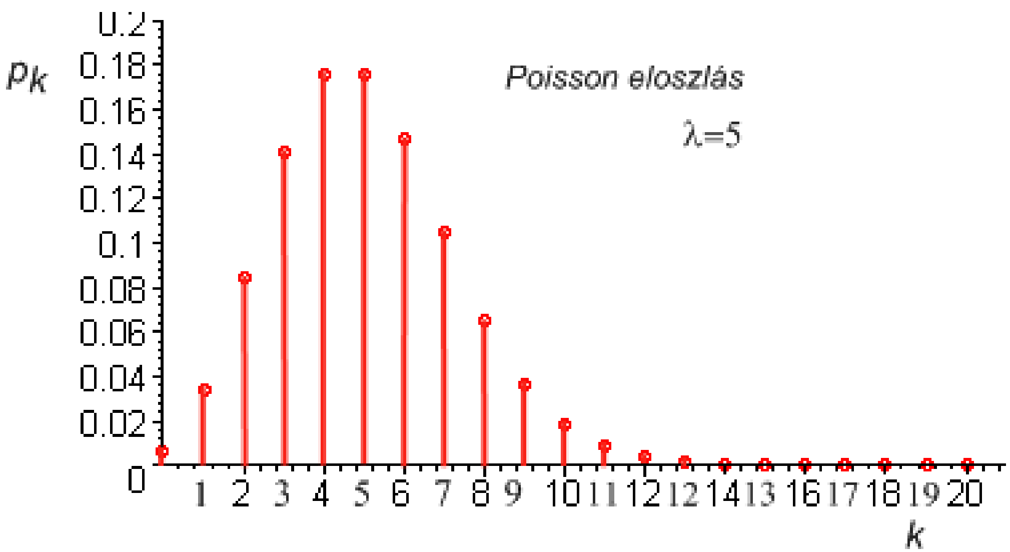
GEOMETRIAI ELOSZLÁS

$$p_k = P(X = k) = p(1-p)^{k-1}, k = 1, \dots n$$



POISSON-ELOSZLÁS

$$p_k = P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, \ldots n, \lambda > 0$$

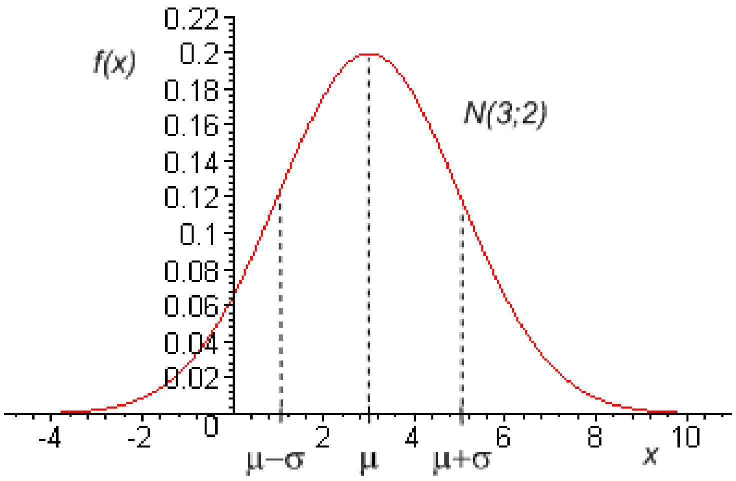


NORMÁLIS ELOSZLÁS

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

$\mu - középérték$

$\sigma - szórás$



Véletlen számok generálása

Middle-square method (J. von Neuman, S. Ulm, M. Metropolis Los Alamos)

0	2041	4165681	1	1656	2742336
2	7423	55100929	3	1009	1018081
4	180	32400	5	324	104976
6	1049	1100401	7	1004	1008016
8	80	6400	9	64	4096
10	40	1600	11	16	256
12	2	4	13	0	0

Egyenletes olosztású véletlen számok generálása

Lineáris kongruenciális generátor

$x_n = (a\, x_{n-1} + b) \bmod m$

m - modulo

a, b, m, x_0 bemeneti paraméterek

ha b=0 multiplikatív generátorról beszélünk

Egyenletes olosztású vélétlen számok generálása

Érvényes:

$$x_n = (a^n x_0 + b(a^n - 1)/(a - 1)) \bmod m$$

Ezután a 0 és 1 közötti véletlen számokat a következő egyenlet alapján készitjük el:

$$r_n = x_n / m$$

Véletlen számok generálása

- Kérdés: $x_n Є\{0, ...,m-1\}$ periódikus sorozat?
- $(x_0=0, a=5, b=1, m=8, x_n= 0, 1, 6, 7, 4, 5, 2, 3, 0)$
- Lineáris kongruenciális generátor

$x_n = (a\, x_{n-1} + b) \bmod m$ m periódikus akkor és csak akkor ha

- A szám amely osztja ***m*** és ***b*** egyenlő 1.
- Ha ***q prímszám***, amely osztja az ***m***, akkor ***q*** osztja az ***a-1*** is.
- Ha ***4*** osztja az ***m***, akkor osztja az ***a-1***.

Példa: $m=2^{32}$, $b=1$, $a=4s+1$, $s>0$ prímszám

Exponenciális eloszlású véletlen számok generálása

Adott az λ>0 és az

eloszlásfüggvény $F(x)=\begin{cases} 1-e^{-\lambda x}, \text{ha } x>0 \\ 0, \text{különben} \end{cases}$

$$x = F^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)$$

$F(x)$ eloszlásfüggvényű valószínűségi változó,

tehát x exponenciális elosztású véletlen szám

Algoritmus MATLAB

- clear all; close all
- lambda=0.5;N=10;
- u=rand(1,N); x=-1/lambda*log(1-u);
- figure(1)
- plot(u,x,'r*')
- xlabel u; ylabel x
- title 'exponenciális eloszlású véletlen számok'

Fibonacci generátor

- Fibonacci számok $x_n = x_{n-1} + x_{n-2}$
- Lagged Fibonacci generator

$x_n = x_{n-j} \bullet x_{n-k} (\mathrm{mod}\ m)$, 0<j<k, m=$2^M$

$\bullet$ – bináris operáció (+,-,*,XOR)

+ *Additive Lagged Fibonacci Generatoror ALFG,*

* *Multiplicative Lagged Fibonacci Generatoror MLFG,*

XOR *generalised feedback shift registeror GFSR*

Mersenne twister $m = 2^{19937} - 1$

Matlab generátorok

- rand – egyenletes elosztás
- randn – normális elosztás
- Randi – egész szákok

rng - Ellenőrzés

• rng(seed)

•rng('shuffle')

•rng(seed, generator)

•rng('shuffle', generator)

•rng('default')

Matlab generátorok

- 'twister': Mersenne Twister
- 'simdTwister': SIMD-oriented Fast Mersenne Twister
- 'combRecursive': Combined Multiple Recursive
- 'multFibonacci': Multiplicative Lagged Fibonacci
- 'v5uniform': Legacy MATLAB® 5.0 uniform generator
- 'v5normal': Legacy MATLAB 5.0 normal generator
- 'v4': Legacy MATLAB 4.0 generator

0-962 számok generálása

- %generating randon numbers between 0-962
- %1 3 7 9 mult 963
- clear all
- close all
- x0=0.12347;N=4000;des=10000;
- x_ran(1)=fix(x0);
- for i=1:N
-     x=(x0-fix(x0))*des*963;
-     x0=x/des;
-     x_ran(i+1)=fix(x0);
- end
- plot(x_ran,'*r')
- hold on
- plot([0 N],[962 962],'g')
- plot([0 N],[990 990],'y')
- hold off

Példák MCM szimulációkra

- A határozott integrál kiszámítása $\int_a^b f(x)dx$
- A π szám meghatározása
- A határozott integrál kiszámítása $\iint_O f(x,y)dxdy$
- Véletlen séta
- Brown mozgás (1,2,3 dimenziós térben)
- Kikötő üzemeltetése

A határozott integrál kiszámítása

$$\int_a^b f(x)dx$$

Bemenet N, a, b, M, f(x)

$m = 0$

i=1, ...,N esetén

Generálás $x_i, y_i, x_i \in \langle a, b \rangle, y_i \in \langle 0, M \rangle$

Kiszámízás $f(x_i)$

Ha $y_i \leq f(x_i)$, m=m+1

$$Felület = \frac{M(b-a)m}{N}$$

A határozott integrál kiszámítása

$$\iint_O f(x,y)\,dxdy$$

Bemenet N, a, b, c, d, M, f(x,y)

$m = 0$

i=1, ...,N esetén

Generálás $x_i, y_i, z_i,$

$x_i \in \langle a, b \rangle, y_i \in \langle c, d \rangle, z_i \in \langle 0, M \rangle$

Kiszámítás $f(x_i, y_i)$

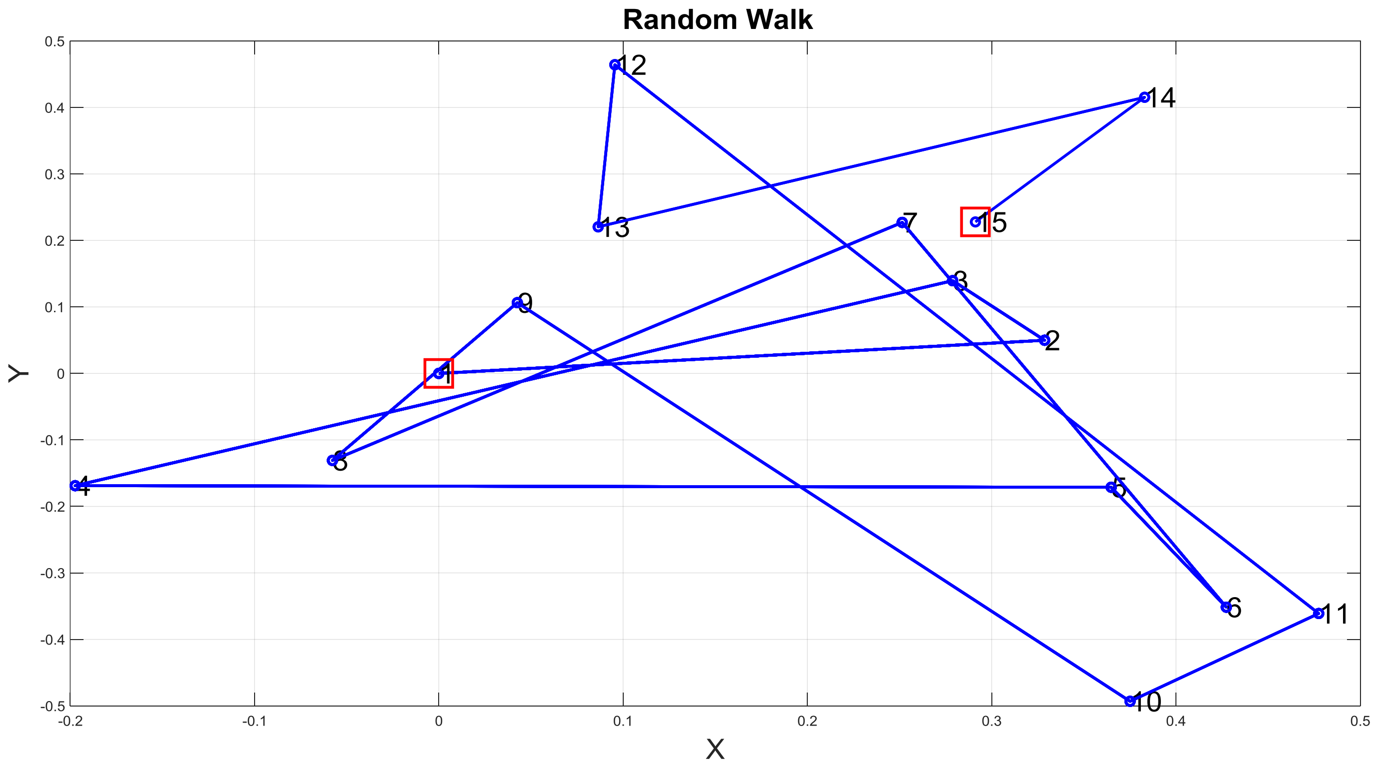
$Ak\ z_i \leq f(x_i, y_i),\ m=m+1$

$$Térfogat = \frac{M(b-a)(d-c)m}{N}$$

Výpočet čísla π

```
% pimc.m
   % Matlab Program to Find Pi using Random Numbers
   % Tom Huber, June 15, 1996
   Nrand = input('How Many Random Numbers ');
   NInside = 0;
   for nloops=1:Nrand
      Xrand = rand;   % Generate Random XY Point
      Yrand = rand;
      Rrand = Xrand^2 + Yrand^2;  % Find its distance from origin
      if (Rrand <= 1)
         NInside = NInside + 1;
      end
   end
   disp(['Total Generated: ' num2str(Nrand) ' Inside Pts: ' ...
      num2str(NInside)]);
   piapprox = 4*NInside/Nrand;
   disp(['  Approximation to pi = ' num2str(piapprox)]);
```

Véletlen séta



Véletlen séta

- deltax = rand(1,numberOfSteps) - 0.5;
- deltay = rand(1,numberOfSteps) - 0.5;
- xy = zeros(numberOfSteps,2);
- for step = 2 : numberOfSteps
- % Walk in the x direction.
- xy(step, 1) = xy(step, 1) + deltax(1,step);
- % Walk in the y direction.
- xy(step, 2) = xy(step, 2) + deltay(1,step);
- % Now plot the walk so far.
- xCoords = xy(1:step, 1);
- yCoords = xy(1:step, 2);
- plot(xCoords, yCoords, 'bo-', 'LineWidth', 2);

Brown mozgás – Brownian motion

- A Brown-mozgás a folyadékban (folyadékban vagy gázban) szuszpendált részecskék véletlenszerű mozgása, amely a gázban vagy folyadékban lévő gyorsan mozgó atomokkal vagy molekulákkal való ütközés eredményeképpen jön létre. Ez a közlekedési jelenség Robert Brown botanikusról kapta a nevét. 1827-ben, miközben mikroszkóppal nézte a vízben lévő pollenszemek belsejében lévő üregekben rekedt részecskéket, észrevette, hogy a részecskék a vízben mozognak; de nem tudta meghatározni a mozgást okozó mechanizmusokat.

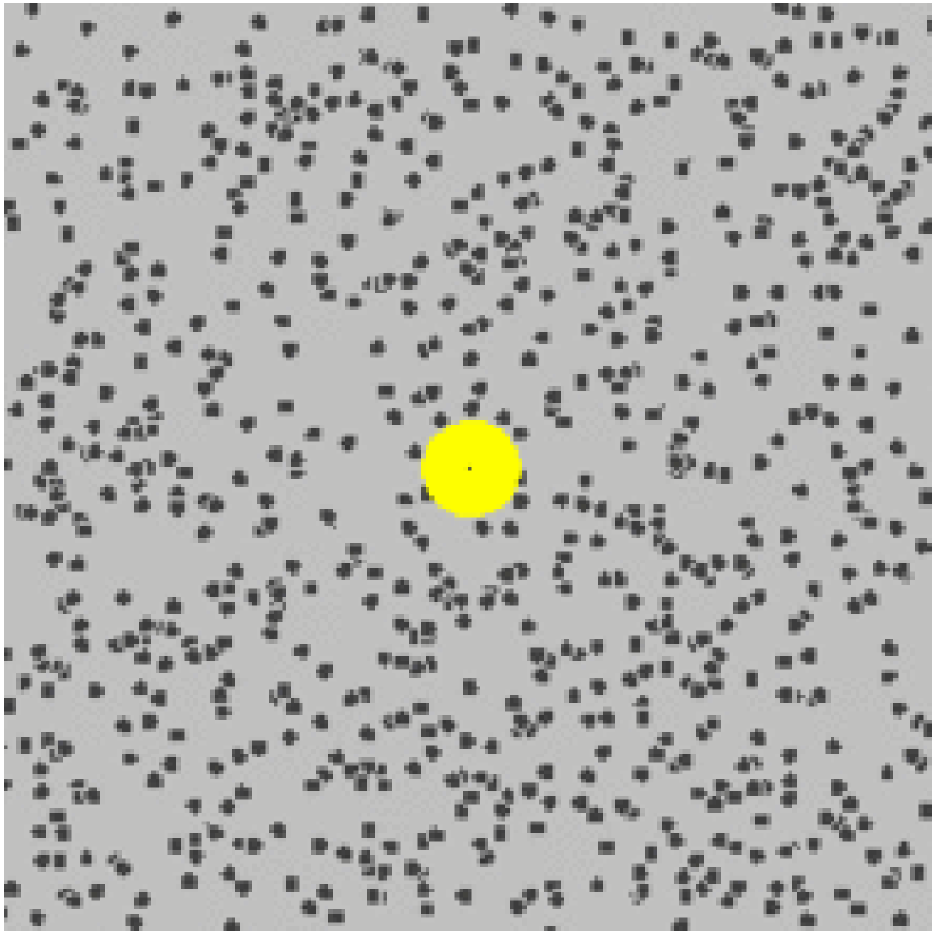
Brown mozgás – Brownian motion

- Az atomokat és molekulákat régóta az anyag alkotóelemeiként képzelték el, és Albert Einstein 1905-ben publikált egy tanulmányt, amelyben részletesen elmagyarázta, hogy a Brown által megfigyelt mozgás miként a virágpor egyes vízmolekulák általi mozgatásának eredménye.

Brown mozgás – Brownian motion

- A Brown-mozgásnak ez a magyarázata meggyőző bizonyítékként szolgált az atomok és molekulák létezésére, amelyet Jean Perrin 1908-ban kísérletileg tovább igazolt. Perrin 1926-ban megkapta a fizikai Nobel-díjat "az anyag nem folytonos szerkezetével kapcsolatos munkájáért". Az atom (részecske) mozgásának iránya folyamatosan változik, és különböző időpontokban a részecskék többet ütköznek az egyik oldalon, mint a másikon, ami a mozgás véletlenszerűnek tűnő természetéhez vezet.

Brown mozgás – Brownian motion



Brown mozgás – Brownian motion

- Egy z változó Brown-mozgást követő változását egy kis \Delta t idő alatt a következő képlet adja meg:

$$\Delta z = \varepsilon \sqrt{\Delta t}$$

- ahol a \varepsilon standardizált normál eloszlású, átlaga 0 és variancia 1.
- És z értékének változása 0 időponttól t-ig a z változásainak összege n \Delta t hosszúságú időintervallumban

$$\Delta t = \frac{t}{n}$$

$$z(t) - z(0) = \sum_{i=1}^{i=n} \varepsilon_i \sqrt{\Delta t}$$

$$\varepsilon_i = (i = 1, 2, 3, \ldots, n)$$

```
t=1; n=500; dt=t/n;
z(1)=0;
for i=1:n
z(i+1)=z(i)+sqrt(dt)*randn;
end

plot([0:dt:t],z)
```

```
t=1; n=500; dt=t/n;
dz=sqrt(dt)*randn(1,n);
z=cumsum(dz);
plot([0:dt:t],[0,z])
```

Generalized Brownian Motion (Generalized Wiener Process)

A variable x following a generalized Brown can be given as

$$dx = a\,dt + b\,dz$$

Where a and b are constants. And, the discrete time model is given by

$$\Delta x = a\Delta t + b\varepsilon\sqrt{\Delta t}$$

Similarly, the change in the value of x from time 0 to t is

$$x(t) - x(0) = a\Delta t + b\sum_{i=0}^{i=n}\varepsilon_i\sqrt{\Delta t}$$

```
t=1; n=1000; dt=t/n;
dz=sqrt(dt)*randn(1,n);
dx=0.4*dt+1.8*dz;
x=cumsum(dx);
plot([0:dt:t],[0,x])
```

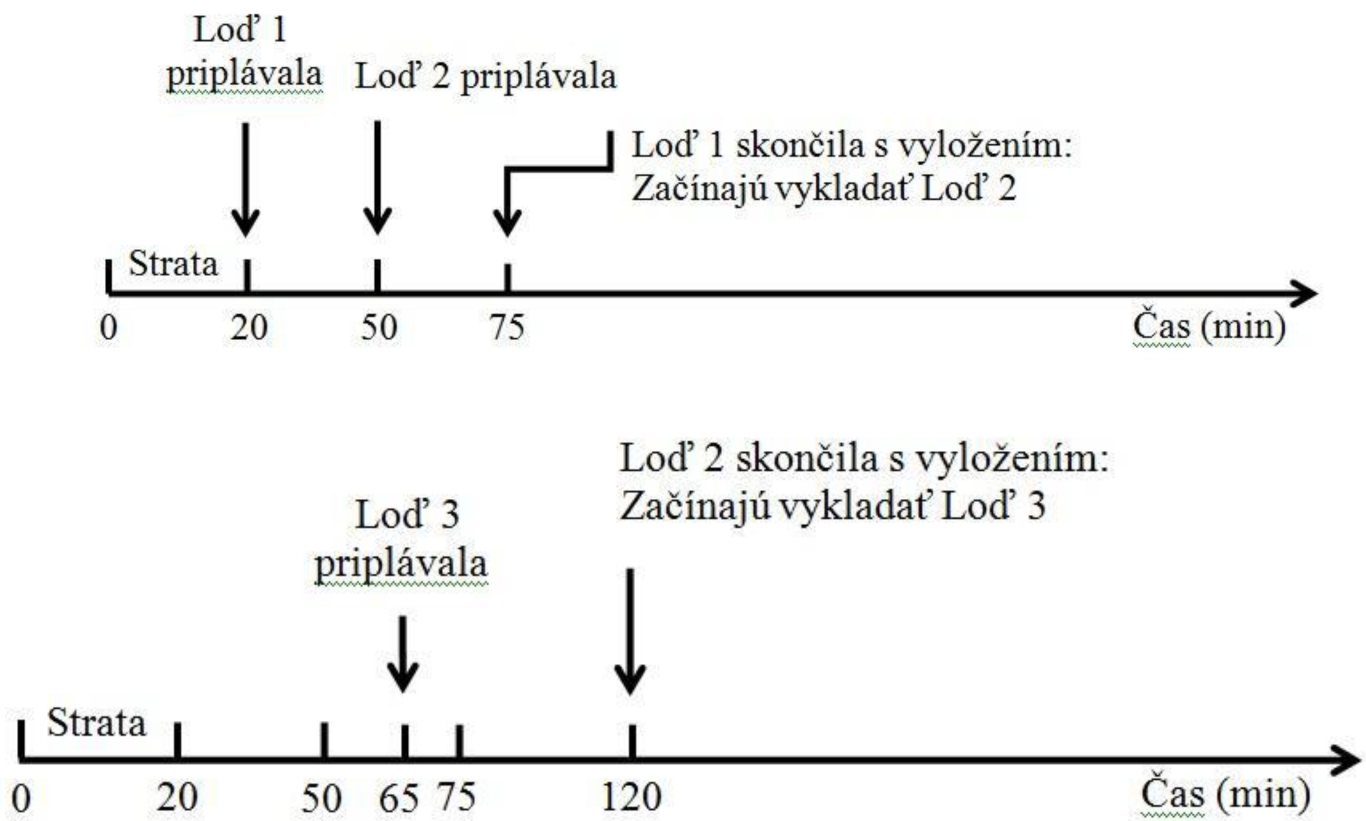
Kikötő modellje

- between_i – az érkezések közötti idő (véletlen szám az <15,145> intervallumból)
- unload_i – kirakodáshoz szükséges idő (véletlen szám az <45,95> intervallumból)
- arrive_i – érkezési idő t=0 kezdettel
- start_i – kirakodás kezdete (t=0)
- finish_i – kirakodás befejezése (t=0)
- wait_i - várakodási idő
- Idle_i – üres kikötő

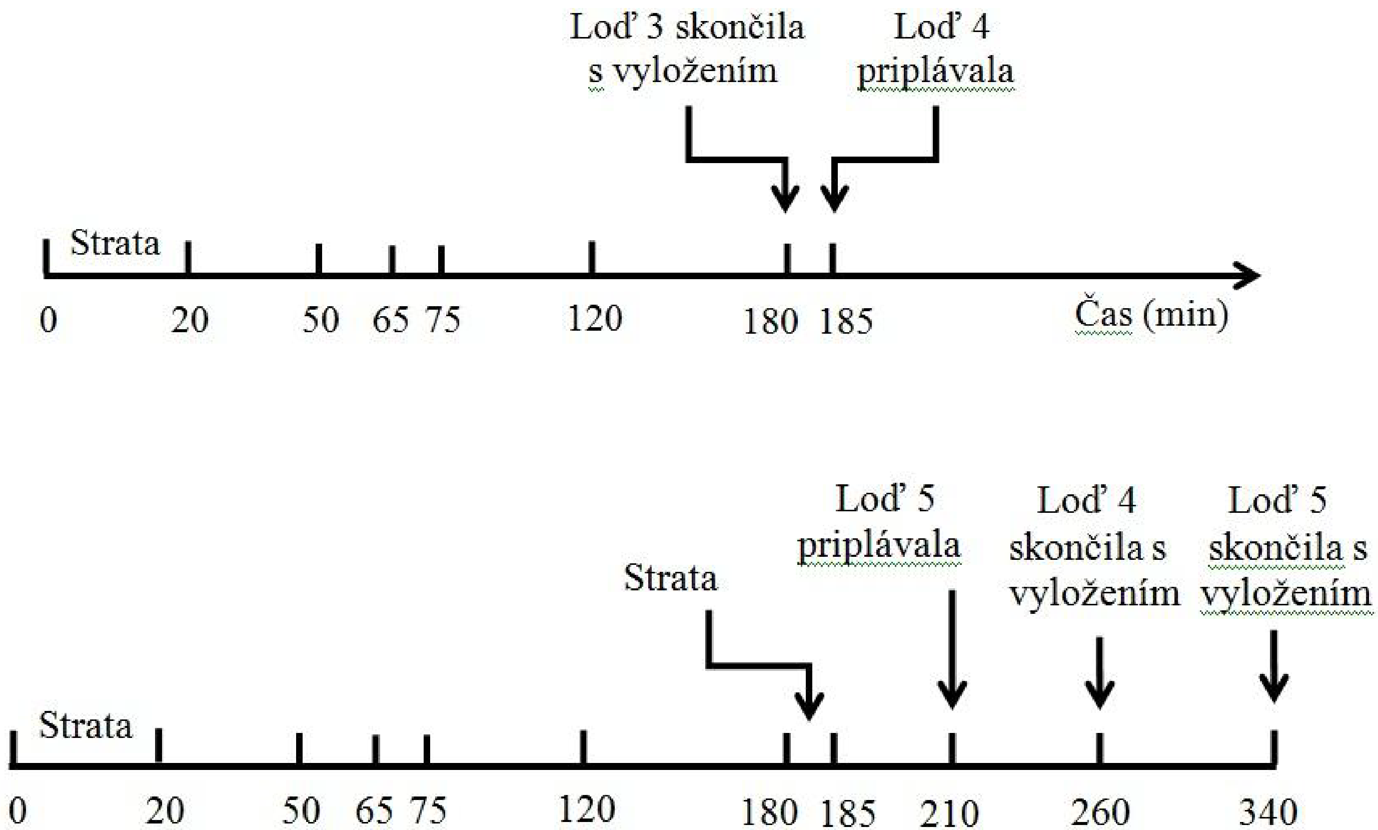
Kikötő modellje

	1. hajó	2. hajó	3. hajó	4. hajó	5. hajó
Érkezések közötti idő	20	30	15	120	25
Kirakodási idő	55	45	60	75	80

Kikötő modellje



Kikötő modellje



Kikötő modellje

- HarTime – átlagidő – kikötőben töltött
- MaxHar – kikötőben töltött maximális időtartalom
- WaiTime – átlagidő-várakozás
- MaxWait – a várakozás maximális időtartalma
- IdleTime_m – átlagidő - üres kikötő
- IdleTime_ – százalék - üres kikötő
- Between – átlagidő - érkezések
- Uload – átlagidő - kirakodás

Algoritmus

- Generálás between(1) a unload(1)
- Tesszük arrive(1)=between(1), start(1)=arrive(1), finish(1)=arrive(1)+unload(1), HarTime=unload(1),MaxHar=unload(1), WaitTime=0,MaxWait=0, IdleTime=arrive(1)

Algoritmus

- i=2,N esetén
- Vétetlen számok generálása between(i) és unload(i)
- arrive(i)=arrive(i-1)+between(i) (következő hajó érkezése)
- timediff=arrive(i)-finish(i-1) (várakozási idő)
- if timediff>=0 idle(i)=timediff, wait(i)=0
- else idle(i)=0, wait(i)=-timediff
- start(i)=arrive(i)+wait(i) (kirakodás kezdete)
- finish(i)=start(i)+unload(i) (kirakodás vége)
- harbor(i)=wait(i)+unload(i) (a kikötőben töltött idő)

- HarTime=HarTime+harbor(i);
- if harbor(i)>MaxHar
- MaxHar=harbor(i);
- end
- WaiTime=WaiTime+wait(i);
- IdleTime=IdleTime+idle(i);
- if wait(i)>MaxWait
- MaxWait=wait(i);
- end

Algoritmus

- Between=Between+between(i);
- Unload=Unload+unload(i);
- finish(N)
- HarTime=HarTime/N;WaiTime=WaiTime/N;IdleTime_p =IdleTime/finish(N);
- Unload=Unload/N;Between=Between/N;IdleTime_m=IdleTime/N;

Kimenet (HarTime, MaxHar, WaiTime, MaxWait, IdleTime_m, Between, Unload, IdleTime_p)