Programování modelů

26. října 2013

1 Spojitý model

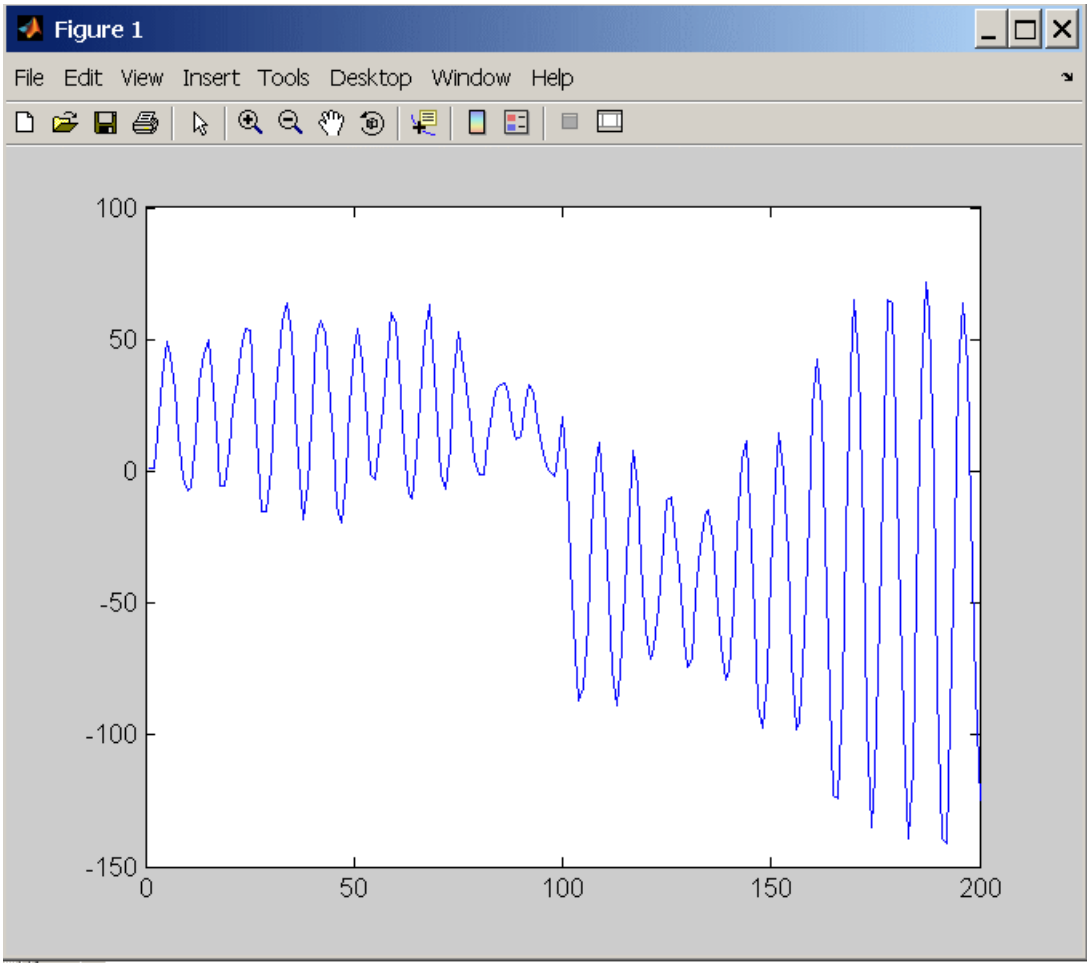
Spojitost modelu říká, že výstup může nabývat libovolné hodnoty z nějakého intervalu. Je to obdoba spojitých rozdělení.

1.1 Lineární regresní model druhého řádu s normálním bílým šumem

Naprogramujte lineární regresní model druhého řádu s normálním šumem $N(0, 20)$. Vektor parametrů $(b_0, a_1, b_1, a_2, b_2, k) = (20; 1,5; -10; -1; 5; -4)$. První dva členy jsou 1 a 1. Řízení je pro prvních 100 kroků +1, pro druhých 100 kroků -1. Vykreslete graf pro prvních 200 členů.

> Všimněte si, jak další členy počítám skalárním součinem vektoru parametrů a regresního vektoru. Je to elegantnější, než tam mít jednu dlouhou a nepřehlednou rovnici. V součinu tudíž NESMÍ být tečka! Druhý z vektorů musí být transponovaný (apostrof), abych měl řádek krát sloupeček. Pokud se mi skalární součin nelíbí, mohu celý vnitřek for cyklu nahradit jedním řádkem:
>
> ```
> y(n)=20*x(n)+1.5*y(n-1)-10*x(n-1)-1*y(n-2)+5*x(n-2)-4+randn*sqrt(20);
> ```
>
> Jinak program vypadá takto:

```
clear all;      %Vyčistí paměť od minulých proměnných
clc;            %Smaže Command Window

x1=ones(1,100);      %100 jedniček
x2=-1*ones(1,100);   %100 mínus jedniček
x=[x1,x2];          %Řízení

y(1)=1;         %První člen
y(2)=1;         %Druhý člen

for n=3:200   %Generuji další členy
    Th=[20,1.5,-10,-1,5,-4];    %vektor parametrů
    Psi=[x(n),y(n-1),x(n-1),y(n-2),x(n-2),1]; %Regresní vektor
    e=randn*sqrt(20);   %Šum
    y(n)=Th*Psi'+e; %Další člen
end;

plot(y);
```

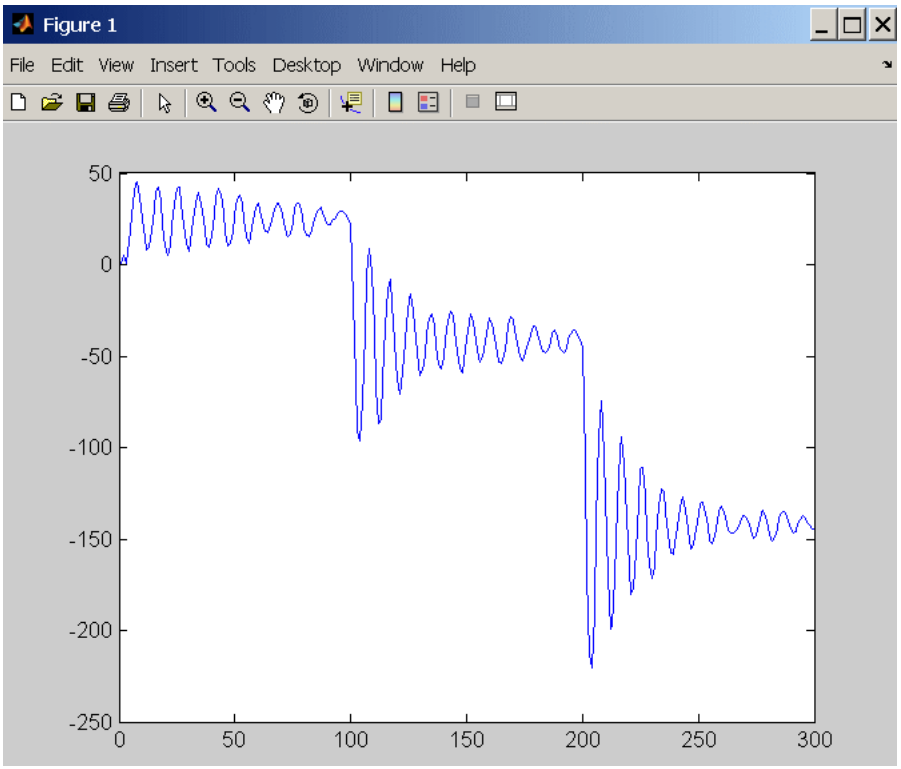


1.2 Model třetího řádu s alternativním bílým šumem

Naprogramujte lineární regresní model třetího řádu s alternativním bílým šumem: $P(1) = 0,5$, $P(-1) = 0,5$. Vektor parametrů $(b_0, a_1, b_1, a_2, b_2, a_3, b_3, k) = (20; 1,5; -10; -1; 5; 0,05, 0, -4)$. První tři členy jsou 1, 5 a 1. Řízení je pro prvních 100 kroků +1, pro druhých 100 kroků -1, pro třetích 100 kroků -4. Vykreslete graf pro prvních 300 členů.

> Všimněte si, že na rozdíl od předchozího příkladu jsem tentokrát vygeneroval celý šum najednou, ještě před for cyklem. Osobně mi přijde generování šumu najednou elegantnější, ale je to celkem jedno.

```
clear all;       %Vyčistí paměť od minulých proměnných
clc;             %Smaže Command Window

%Řízení
x1=ones(1,100);       %100 jedniček
x2=-1*ones(1,100);    %100 mínus jedniček
x3=-4*ones(1,100);    %100 mínus čtyřek
x=[x1,x2,x3];         %Řízení

%Šum
e=rand(1,300);   %300 hodnot 0...1
e=2*e;           %0...2
e=fix(e);        %0, 1
e=2*e;           %0, 2
e=e-1;           %-1, 1

y(1)=1;          %První člen
y(2)=5;          %Druhý člen
y(3)=1;          %Třetí člen

for n=4:300   %Generuji další členy
    Th=[20,1.5,-10,-1,5,0.05,0,-4];    %vektor parametrů
    Psi=[x(n),y(n-1),x(n-1),y(n-2),x(n-2),y(n-3),x(n-3),1]; %Reg.vek.
    y(n)=Th*Psi'+e(n); %Další člen
end;

plot(y);
```



2 Diskrétní model

V diskrétních modelech může výstup nabývat jen jedné z několika možných hodnot. Je to obdoba diskrétních rozdělení.

Zatímco ve spojitých modelech se náhodnost do modelu dostává přidáváním náhodných členů - bílých šumů, v diskrétních modelech se náhodnost objevuje jiným způsobem. Diskrétní model udává pravděpodobnosti té či oné hodnoty a na základě těchto pravděpodobností se pak následující výstup náhodně generuje.

2.1 Diskrétní model prvního řádu s jedním řízením

Naprogramujte diskrétní model prvního řádu s řízením s následující tabukou podmíněných pravděpodobností:

x_n	y_{n-1}	$P(y_n = 1)$	$P(y_n = 2)$
1	1	1	0
1	2	0	1
2	1	0	1
2	2	1	0
3	1	0,5	0,5
3	2	0,5	0,5

Vykreslete 40 členů posloupnosti. Prvních deset hodnot řízení dejte 1, druhých deset 2, třetích deset 3 a čtvrtých deset 1. První člen posloupnosti je 1.

> Všimněte si, že pro řízení 1 posloupnost zachovává hodnoty, pro řízení 2 je střídá a pro řízení 3 je generuje zcela nezávisle na předchozím členu.

```
clear all;       %Vyčistí paměť od minulých proměnných
clc;             %Smaže Command Window

%Řízení
x1=ones(1,10);        %10 jedniček
x2=2*ones(1,10);      %10 dvojek
x3=3*ones(1,10);      %10 trojek
x=[x1,x2,x3,x1];         %Řízení

%Tabulka podmíněných pravděpodobností
Tab=[1,   0;
     0,   1;
     0,   1;
     1,   0;
     0.5, 0.5;
     0.5, 0.5];
```

```
%Tabulka distribučních funkcí
DF=cumsum(Tab,2);

y(1)=1;          %První člen

for n=2:40                    %Generuji další členy
    i=2*(x(n)-1)+y(n-1);     %volba řádku tabulky
    y(n)=sum(rand>DF(i,:))+1;
end;

plot(y,'r-*');
```

Nejprve jsme si připravili řízení. Pak jsme zadali pravděpodobnostní tabulku. Každý řádek je pravděpodobnostní funkcí. Z pravděpodobnostních funkcí jsme udělali distribuční funkce pomocí funkce *cumsum*. Parametr 2 zařídí, že kumulativní sčítání proběhne v řádcích a nikoliv ve sloupcích.

Zadali jsme první člen a for cyklus nastavili až od 2.

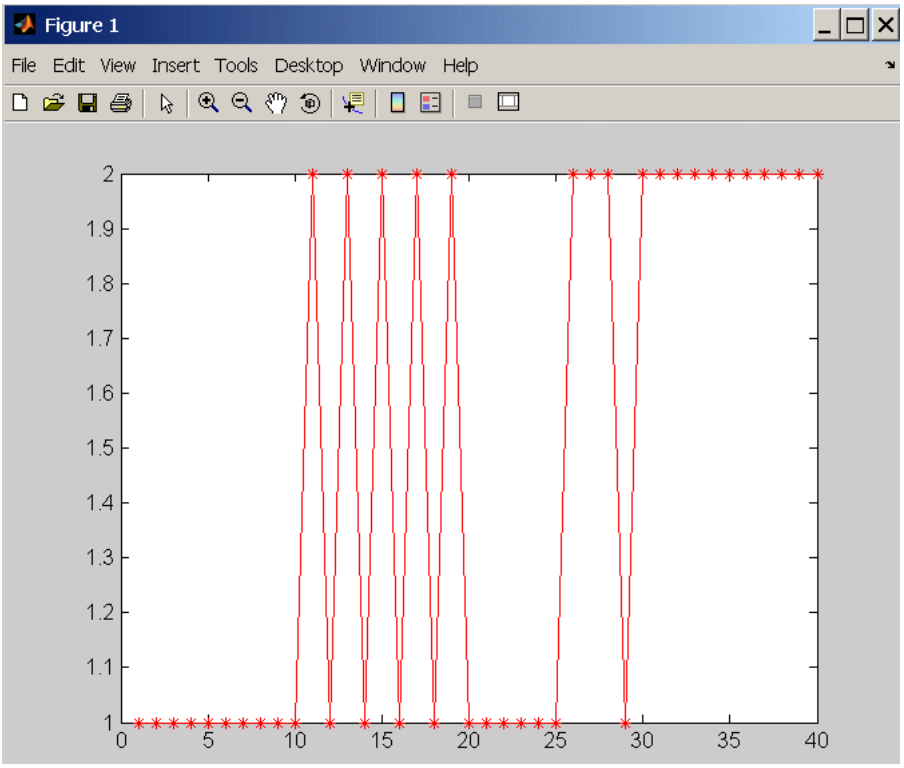
Uvnitř for cyklu nejprve zjistíme relevantní řádek tabulky. Dvojka za rovnítkem znamená, že se x_n mění vždy po dvou řádcích. Z prvního členu tedy dostaneme 0, 2 nebo 4 (podle řízení) a přičteme 1 nebo 2 podle y_{n-1}.

Následuje tajuplný řádek, kdy generujeme y_n. Zápis *DF(i,:)* znamená i-tý řádek tabulky DF, všechny sloupečky. Tento řádek je distribuční funkcí, tedy rostoucí posloupností čísel, kde poslední číslo je jedna. Příkaz *rand* vygeneruje náhodné číslo mezi 0 a 1 a podmínka *rand>DF(i,:)* se vyhodnotí pro každou položku vektoru *DF(i,:)* zvlášť. Získáme tak vektor jedniček a nul, podle toho, jestli je podmínka splněna nebo ne. Funkce *sum* sečte počet jedniček a po přičtení 1 dostaneme příslušné y_n.

Předveďme si to na příkladu. Uvažujme řádek tabulky: 0,5 - 0,5 a nechť funkce *rand* vygeneruje číslo 0,3. Mělo by to fungovat tak, že pokud funkce *rand* vygeneruje číslo do poloviny, bude jednička. Pokud přes polovinu, bude dvojka. Funkce *cumsum* nám dá příslušný řádek DF: 0,5 - 1. Podmínka *rand>DF(i,:)* tedy není splněna nikdy a dostaneme vektor 0 - 0. Funkce *sum* tedy dá nulu. Po přičtení 1 dostáváme opravdu jedničku.

Zkusme to samá s náhodným číslem 0,7. Podmínka *rand>DF(i,:)* bude splněna pro první člen, pro druhý nikoli. Dostaneme tedy vektor 1 - 0. Funkce *sum* dá jedničku. Po přičtení 1 dostáváme dvojku.

Graf funkce bude vypadat nějak takto:



Vidíme, že mezi 1 a 10 graf opravdu drží hodnotu, mezi 10 a 20 ji pravidelně střídá, hodnoty mezi 20 a 30 jsou náhodně generované a po 30 se opět drží hodnota. Tak, jak říkala tabulka.

2.2 Diskrétní model prvního řádu s řízením prvního řádu

Naprogramujte diskrétní model prvního řádu s řízením s následující tabukou podmíněných pravděpodobností:

x_{n-1}	x_n	y_{n-1}	$P(y_n = 1)$	$P(y_n = 2)$
1	1	1	1	0
1	1	2	0	1
1	2	1	0	1
1	2	2	1	0
2	1	1	1	0
2	1	2	0	1
2	2	1	1	0
2	2	2	0	1

Vykreslete prvních 40 členů posloupnosti. Řízení bude jednička kromě 5. až 10. členu, 20. členu a 25. až 35. členu. V těchto případech bude řízení dvojka. První člen posloupnosti je 2.

Zajímavý je třetí a čtvrtý řádek tabulky, který říká, že pokud v řízení po jedničce přijde dvojka, výstup se přepne. Ve všech ostatních případech se výstup zachovává. Můžeme tedy očekávat přepnutí na páté, dvacáté a dvacáté páté pozici.

Můžeme si také všimnout, že tato posloupnost vlastně není stochastická, neboť tabulka obsahuje jen nuly a jedničky a žádná náhoda v ní tudíž nefiguruje.

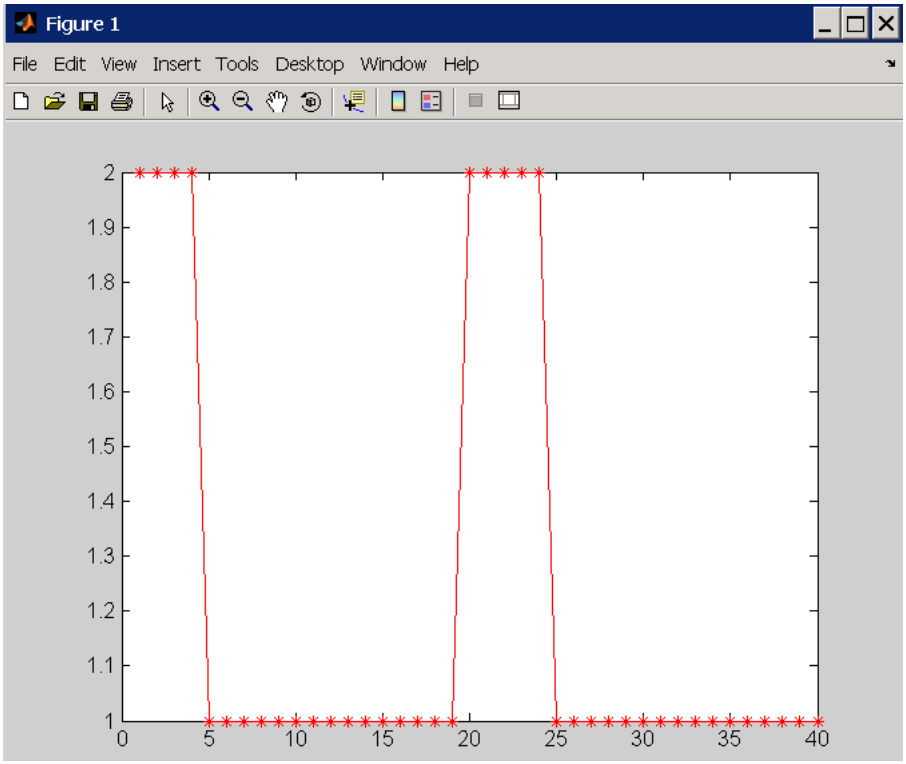
```
clear all;       %Vyčistí paměť od minulých proměnných
clc;             %Smaže Command Window

%Řízení
x=ones(1,40);              %40 jedniček
x(5:10)=2*ones(1,6);       %6 dvojek
x(20)=2;                   %jedna dvojka
x(25:35)=2*ones(1,11);     %11 dvojek

%Tabulka podmíněných pravděpodobností
Tab=[1,   0;
     0,   1;
     0,   1;
     1,   0;
     1,   0;
     0,   1;
     1,   0;
     0,   1];
```

```
%Tabulka distribučních funkcí
DF=cumsum(Tab,2);

y(1)=2;          %První člen

for n=2:40                    %Generuji další členy
    i=4*(x(n-1)-1)+2*(x(n)-1)+y(n-1);%volba řádku
    y(n)=sum(rand>DF(i,:))+1;
end;

plot(y,'r-*');
```

Všimněte si, jak je zadáno řízení.

Jinak je hlavní změnou složitější řádek pro výpočet i. Čtyřka znamená, že se x_{n-1} mění po čtyřech řádcích. Dvojka znamená, že se x_n mění po dvou řádcích. Podobně bychom zvládli i mnohem složitější tabulky.

Získáme následující graf:



Opravdu vidíme, že ke změně dochází na páté, dvacáté a dvacáté páté pozici, jak jsme čekali. Nezáleží přitom na délce sekvence dvojek v řízení.

3 Logistický model

Logistický model, jak název napovídá, souvisí s jednou základní úlohou z dopravy: Když mám tolik a tolik času a podmínky jsou takové a takové, stihnu to? Jeho použití je však mnohem širší.

Používá a všude tam, kde podmínky jsou alespoň z části spojité a výstup je dvouhodnotový. Např. stihnu/nestihnu.

Pokud by doba dojezdu měla za daných podmínek normální rozdělení, měli bychom ve výpočtu použít distribuční funkci normálního rozdělení. Často se však místo ní používá logistická funkce, která má podobný průběh. Poznamenávám, že střední hodnoty normálního rozdělení $N(0,1)$ i logistického rozdělení jsou nula, avšak rozptyl logistického rozdělení není jedna.

Náhoda se v modelu objevuje jednak pomocí generování členu posloupnosti na základě spočtené pravděpodobnosti, jednak v rovnici pro z mohou vystupovat šumy. Logistický model je jakýmsi sloučením spojitého a diskrétního modelu.

3.1 Logistický model nultého řádu se spojitým řízením

Naprogramujte logistický model definovaný logistickou funkcí $p(z_n) = \frac{e^{z_n}}{1+e^{z_n}}$ a vztahem $z_n = 0,1 \cdot (x_n - 60)$. Řízení volte od 20 do 100 po jedné. Vykreslete graf výstupů v závislosti na řízení.

Tato úloha může odpovídat cestě s potřebným středním časem 60 minut. Výstup jednička znamená stihli jsme, výstup nula nestihli jsme. Veličina x udává, kolik času jsme měli, když jsme vyrazili.

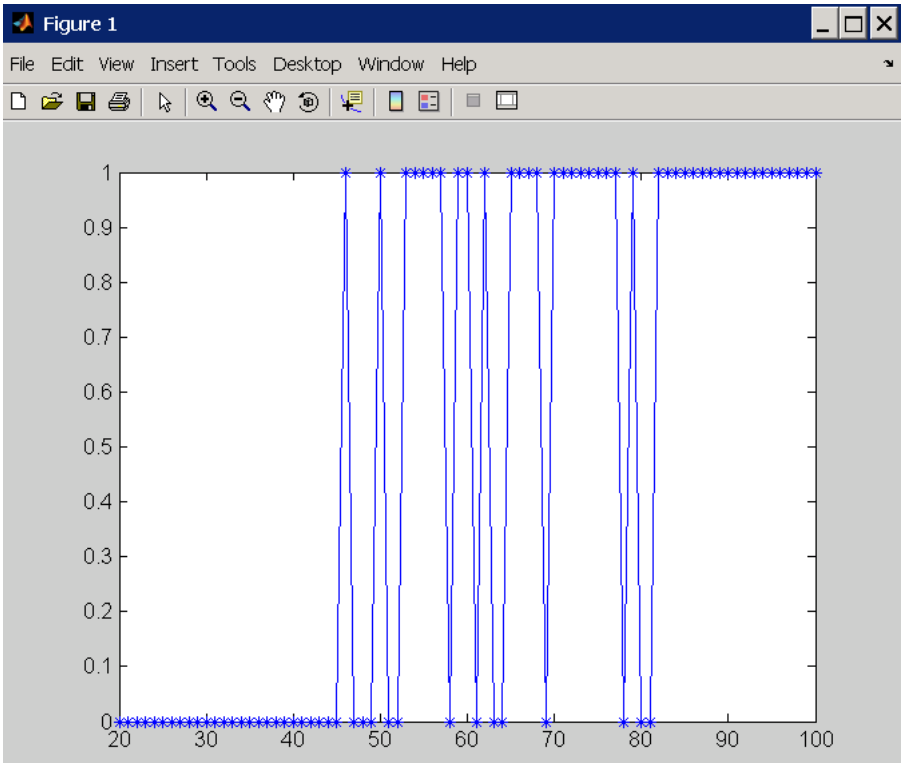
Pro $x = 60$ je z nula a hodnota $p(z)$ tedy 0,5. Pro větší x bude pravděpodobnost jedničky stále větší.

Program je následující:

```
clear all;
clc;

x=20:1:100; %Řízení

for n=1:81  %Počet členů řízení
    z(n)=0.1*(x(n)-60);
    p(n)=exp(z(n))./(1+exp(z(n)));
    y(n)=(rand<p(n));
end;

plot(x,y,'-*');
```

Rozmyslete si, jak funguje řádek generující y_n.

Získáme následující graf:



Vidíme, že kolem 45. minuty se nám objevil první stihnutý případ. Výskyt úspěšných případů je čím dál větší, až po 82. minutě už byly úspěšné všechny.

3.2 Logistický model prvního řádu se spojitým řízením

Naprogramujte logistický model definovaný logistickou funkcí $p(z_n) = \frac{e^{z_n}}{1+e^{z_n}}$ a vztahem $z_n = 0,1 \cdot (x_n + 10 \cdot y_{n-1} - 60)$.
Řízení volte od 100 do 20. Prní člen je 1. Vykreslete graf výstupů v závislosti na řízení.

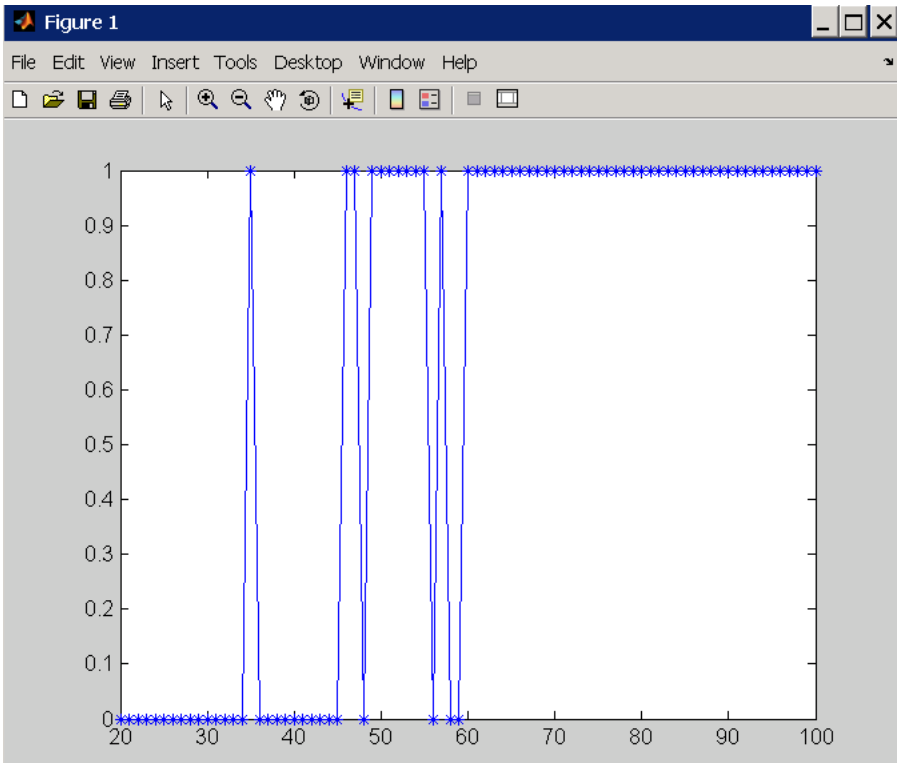
Přidaný člen v rovnici pro z může reprezentovat např. zvýšení elánu po úspěšném pokusu a větší snahu dosáhnout opět úspěchu.

```
clear all;
clc;

%První případ
x=100:-1:20; %Řízení

y(1)=1; %První člen

for n=2:81  %Počet členů řízení
    z(n)=0.1*(x(n)+10*y(n-1)-60);
    p(n)=exp(z(n))./(1+exp(z(n)));
    y(n)=(rand<p(n));
end;

plot(x,y,'b-*');
```

Všimněte si mínus jedničky v řádku s řízením. Je to takový chyták.

Protože máme model prvního řádu, musíme zadat první člen a for cyklus mít až od dvojky.

Získáme následující graf:



Závodník zkoušel stále kratší časy od stovky k dvacítce. Vidíme, že motivace získaná předchozím úspěšným pokusem udělala své a až do 60. minuty stíhal.