Skalarni produkt in korelacija

Jure Aplinc

19.4.2009

Povzetek

V fiziki je uporaba skalarnega prodikta zelo pogosta saj imamo skoraj vedno opravka z vektorskimi količinami. Skalarni produkt je invariantna operacija med vektorskimi prostori z skalarnim produktom.Paru funkcij, ki sta v Hilberovem prostoru lahko priredimo skalarni produkt, ki je v tem primeru definiran kot integral zmnožka teh dveh funkcij.

1 Naloga

V najnovejši številki Obzornika je objavljen zanimiv članek o miniaturni magnetni črpalki. Avtorji napovedo linearno zvezo med frekvenco rotorja in hitrostjo toka; meritve v datoteki "HitrostTokaOdFrekvence.txt"to potrjujejo. Določi korelacijski koeficient zveze med obema količinama.

1.1 potek reševanja

Za računanje korelacijskega koeficienta sem napisal program v jeziku C.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
int main(void){
FILE *fin;
int i;
double ypov, xpov, ysigma, xsigma, sum, sum2, sum3, yi, xi, a, n, skalab, r, R, e;
//sigmay (1 pass)
fin=fopen("frekvence.dat", "r");
sum2=0;
sum=0;
n=0;
yi=0;
ypov=0;
ysigma=0;
while(fscanf(fin, "%lf %lf %lf", &a, &yi, &e)==3){
yi=yi;
sum+=yi;
sum2+=yi*yi;
n+=1;
}
ypov=sum/n;
ysigma=sqrt(sum2/n-ypov*ypov);
printf("(one pass) ysigma=%g ypov=%g N=%g\n", ysigma, ypov, n);
fclose(fin);
//sigmax (1 pass)
```

```
fin=fopen("frekvence.dat", "r");
sum2=0;
sum=0;
n=0;
xi=0;
xpov=0;
xsigma=0;
while(fscanf(fin, "%lf %lf %lf", &xi, &yi, &e)==3){
xi=xi;
sum+=xi;
sum2+=xi*xi;
n+=1;
}
xpov=sum/n;
xsigma=sqrt(sum2/n-xpov*xpov);
printf("(one pass) xsigma=%g xpov=%g N=%g\n", xsigma, xpov, n);
fclose(fin);
//r(a, b)
fin=fopen("frekvence.dat", "r");
skalab=0;
n=0;
r=0;
xi=0;
yi=0;
while(fscanf(fin, "%lf %lf %lf", &xi, &yi, &e)==3){
skalab+=xi*yi;
n++;
}
r=skalab/n;
printf("r=%lf, slalab=%lf\n", r, skalab);
R=(r-xpov*ypov)/(xsigma*ysigma);
printf("R=%lf\n", R);
fclose(fin);
return 0;

}
```

1.2 rešitev

	R (korelacijski koeficient)
Brez upoštevanja napake	0.987970
Napaka prišteta vsem hitrostim	0.987949
Napaka odšteta vsem hitrostim	0.987975
Napaka izračunana po formuli	0.987970 +− 0.17128

Komentar k rezultatom:

Vidimo lahko, da se korelacija le malo spremeni, če upoštevamo napako, kot, da je ta pozitivna ali negativna za vse izmerke.

2 Naloga

Ameriška uprava za zdravila (FDA – Food and Drug Administration) je preskusila čudežno zdravilo mirabilitin za zvonjenje v ušesih (tintinabulus). V datoteki "Tintin.datšo podani rezultati dvojno

slepega preskusa. Določi korelacijski koeficient med dozo (v mg/kg žive mase) in stanjem bolezni po terapiji (ur zvonjenja na teden).

2.1 potek reševanja

Uporabil sem podoben program kot pri nalogi 1 in izračunal korelacijo.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
int main(void){
FILE *fin;
int i;
double ypov, xpov, ysigma, xsigma, sum, sum2, sum3, yi, xi, a, n, skalab, r, R, e;
//sigmay (1 pass)
fin=fopen("Tintin.dat", "r");
sum2=0;
sum=0;
n=0;
yi=0;
ypov=0;
ysigma=0;
while(fscanf(fin, "%lf %lf %lf", &a, &xi, &yi)==3){
yi=yi;
sum+=yi;
sum2+=yi*yi;
n+=1;
}
ypov=sum/n;
ysigma=sqrt(sum2/n-ypov*ypov);
printf("(one pass) ysigma=%g ypov=%g N=%g\n", ysigma, ypov, n);
fclose(fin);
//sigmax (1 pass)
fin=fopen("Tintin.dat", "r");
sum2=0;
sum=0;
n=0;
xi=0;
xpov=0;
xsigma=0;
while(fscanf(fin, "%lf %lf %lf", &a, &xi, &yi)==3){
sum+=xi;
sum2+=xi*xi;
n+=1;
}
xpov=sum/n;
xsigma=sqrt(sum2/n-xpov*xpov);
printf("(one pass) xsigma=%g xpov=%g N=%g\n", xsigma, xpov, n);
fclose(fin);
//r(a, b)
fin=fopen("Tintin.dat", "r");
skalab=0;
n=0;
```

```
r=0;
xi=0;
yi=0;
while(fscanf(fin, "%lf %lf %lf", &a, &xi, &yi)==3){
skalab+=xi*yi;
n++;
}
r=skalab/n;
printf("r=%lf, skalab=%lf\n", r, skalab);
R=(r-xpov*ypov)/(xsigma*ysigma);
printf("R=%lf\n", R);
fclose(fin);
return 0;

}
```

2.2 rešitev

R (korelacijski koeficient)
-0.394090

Slika 1: Število ur zvonenja v odvisnosti od doze zdravila!

Komentar:

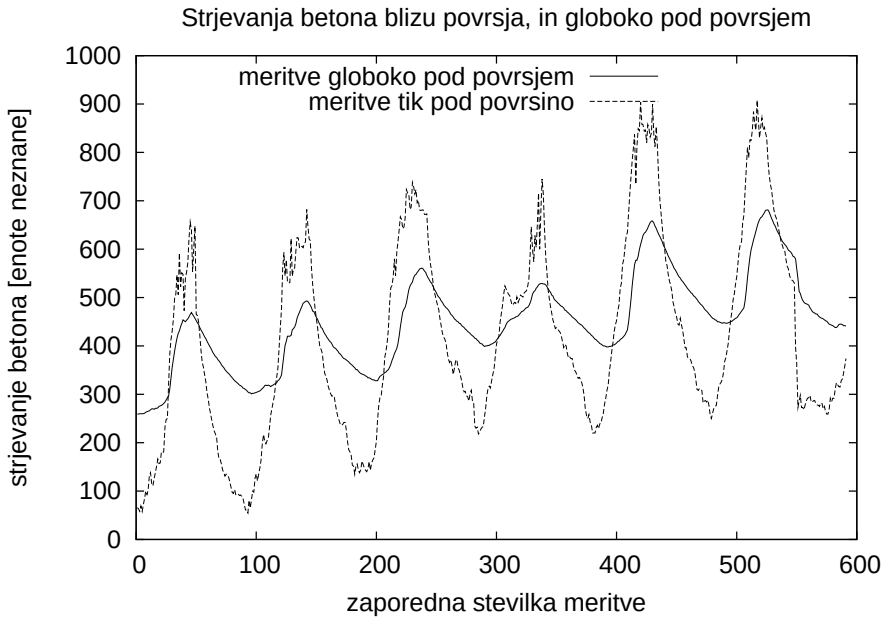
kot vidimo z grafa nekaj pacijentov ni dobilo zdravila. Tisti, ki pa so prejeli zdravilo jim je zvonilo v glavi manj ur na teden poseej če so vzeli več zdravila. zato je R negativen. Zaradi velike razsutosti podatkov je odvisnost medla in R približno 0.4, kar je dovolj, da potrdijo zdravilo.

3 Naloga

Pred leti smo v okviru mednarodnega projekta v našem znanem gradbenem podjetju merili hitrost strjevanja betona. Ulili so nekaj metrov velik betonski blok, v katerega je bila vdelana cela vrsta termočlenov za sprotno merjenje temperature. Datoteka "Beton.dat"podaja izmerke v razdobju šestih dni v dveh merilnih točkah. Prva je blizu površine, druga globoko v notranjosti. (Prvi stolpec je zaporedna številka meritve – časovni interval med njimi lahko oceniš iz očitnih dnevnih nihanj temperature.) Določi efektivno zakasnitev med obema signaloma iz njune korelacijske funkcije.

3.1 numerično računanje korelacijske funkcije

Za začetek poglejmo kakšna grafa da meritev:



Slika 2: Hitrost strjevanja betona v dveh točkah!

Iz grafa je očitno in zelo enostavno ugotoviti, da se signal spreminja peeriodično. Periodo lahko odčitamo iz grafa samega in natančneje iz datoteke podatkov.

Na en dan odpade 96 meritev, kar pomeni eno meritev vsakih 15 nimut.

Za izračun korelacijske funkcije sem napisal program v jeziku C.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
double correlate(double start, double all){
FILE *fin, *fout1, *fout2, *fin1, *fin2;
int i, i1, i2;
double yi, xi, a, n, skala, r, e;
fin=fopen("Beton.dat", "r");
skala=0;
n=0;
r=0;
```

```
xi=0;
yi=0;
//datotečne operacije
fout1=fopen("core1.out", "w");
fout2=fopen("core2.out", "w");
for(i=1; i<=all; i++){
fscanf(fin, "%lf %lf %lf", &a, &xi, &yi);
fprintf(fout1, "%d %lf\n", i, xi);
fprintf(fout2, "%d %lf\n", i, yi);
}
fclose(fout2);
fclose(fout1);
fclose(fin);
//zdaj so kolone ločene
//sledi skalarni produkt
fin=fopen("Beton.dat", "r");
fin1=fopen("core1.out", "r");
fin2=fopen("core2.out", "r");
i1=0;
for(i=1; i<=all; i++){
fscanf(fin1, "%d %lf", &i1, &xi);
if(i1>start){
fscanf(fin2, "%d %lf", &i2, &yi);
skala+=xi*yi;
}
}
printf("skala=%lf", skala);
fclose(fin2);
fclose(fin1);
fclose(fin);
r=skala/(all-start);
return r;
}
int main(void){
FILE *fin, *fout3;
fin=fopen("Beton.dat", "r");
fout3=fopen("core3.out", "w");
double n, a, xi, yi, all, start, r;
start=0;
all=0;
while(fscanf(fin, "%lf %lf %lf", &a, &xi, &yi)==3){
all++;
}
fclose(fin);
printf("all=%lf", all);
fin=fopen("Beton.dat", "r");
while(fscanf(fin, "%lf %lf %lf", &a, &xi, &yi)==3){
r=correlate(start, all);
printf("r=%lf\n", r);
fprintf(fout3, "%lf %lf\n", start, r);
start++;
}
fclose(fin);
fclose(fout3);
```

```
return 0;
}
```

Program vrne tabelirano korelacijsko funkcijo, ki jo prikazuje naslednji graf:

Slika 3: Korelacjka funkcija strjevanja betona za dve točki (v notranjosti in tik pod površino bloka). Korelacijska funkcija je narisana za primer, ko prvi signal premikamo po drugem in računamo korelacijski koeficient in nato še če drug signal pomikamo po prvem.

Iz grafa korelacijske funkcije je očitno, da se morata signala medsebojno premakniti, da se prekrijeta in - takrat ima korelacijska funkcija maksimum.

Iz grafa torej odčitamo (pomagamo pa si še z zabelo obdelanih podatkov), da je potrebno funkciji zamakniti za 9 enot (+n*en dan) da nastopi maksimum.

Ta zamik ustreza premiku 2 uri 15 minut!

Komentar k korelacijski funkciji:

Iz zgornjega grafa je vidno, da se krivulja na obeh straneh strmo spusti to je posledica tega, da takrat v preseku signalov ostanejo le najbolj nizke točke.

4 Naloga

V datoteki "Luna.efe"je dana efemerida Lune za eno od preteklih let. Stolpci so: dan začenši s 1.1., nato rektascenzija (nebesna dolžina) v urah in minutah, in nazadnje deklinacija (nebesna širina) v stopinjah, ob 0h svetovnega časa tega dne. Iz avtokorelacijske funkcije deklinacije čim bolj natančno določi Lunino periodo tira. (Lahko si pomagaš z odvajanjem.)

4.1 Potek reševanja

Najprej sem datume spremenil v številke dnevov (1 začenši z 1.1). Nato sem napravil sem program precej podoben tistemu pri nalogi 3, ki računa avtokorelacijsko funkcijo:

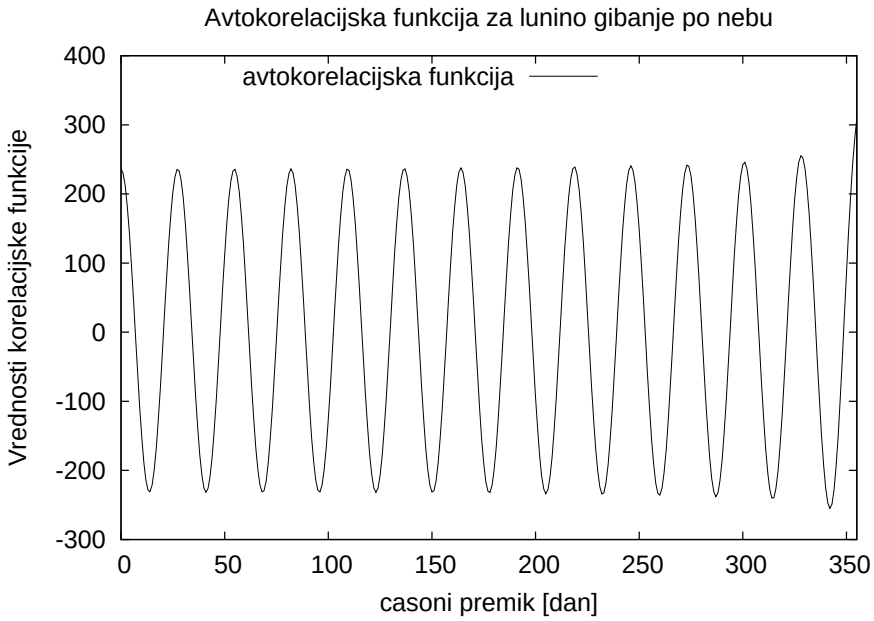
```
#include<stdio.h>
#include<stdlib.h>
```

```
#include<math.h>
#include<string.h>
double correlate(double start, double all){
FILE *fin, *fout1, *fout2, *fin1, *fin2;
int i, i1, i2;
double yi, xi, a, n, skala, r, e;
//sledi skalarni produkt
fin=fopen("luna.dat", "r");
fin1=fopen("luna1.out", "r");
fin2=fopen("luna2.out", "r");
i1=0;
skala=0;
for(i=1; i<=all; i++){
fscanf(fin1, "%d %lf", &i1, &xi);
if(i1>start){
fscanf(fin2, "%d %lf", &i2, &yi);
skala+=xi*yi;
}
}
printf("skala=%lf", skala);
fclose(fin2);
fclose(fin1);
fclose(fin);
r=skala/(all-start);
return r;
}
int main(void){
FILE *fin, *fout3;
fin=fopen("luna1.out", "r");
fout3=fopen("gagarin.out", "w");
double n, a, xi, yi, all, start, r;
start=0;
all=0;
while(fscanf(fin,"%lf %lf",&xi, &yi)==2){
all++;
}
fclose(fin);
printf("all=%lf", all);
fin=fopen("luna1.out", "r");
while(fscanf(fin, "%lf %lf", &xi, &yi)==2){
r=correlate(start, all);
printf("r=%lf\n", r);
fprintf(fout3, "%lf %lf\n", start, r);
start++;
}
fclose(fin);
fclose(fout3);
return 0;
}
```

Program deluje podobno kot prejšnji in sicer zamika en (tokrat isti) stolpec ob drugem in računa skalarne produkte zamaknjenih funkcij, ki jih nato deli z številom točk, ki padejo v prekrivajoči interval.

4.2 Rešitev

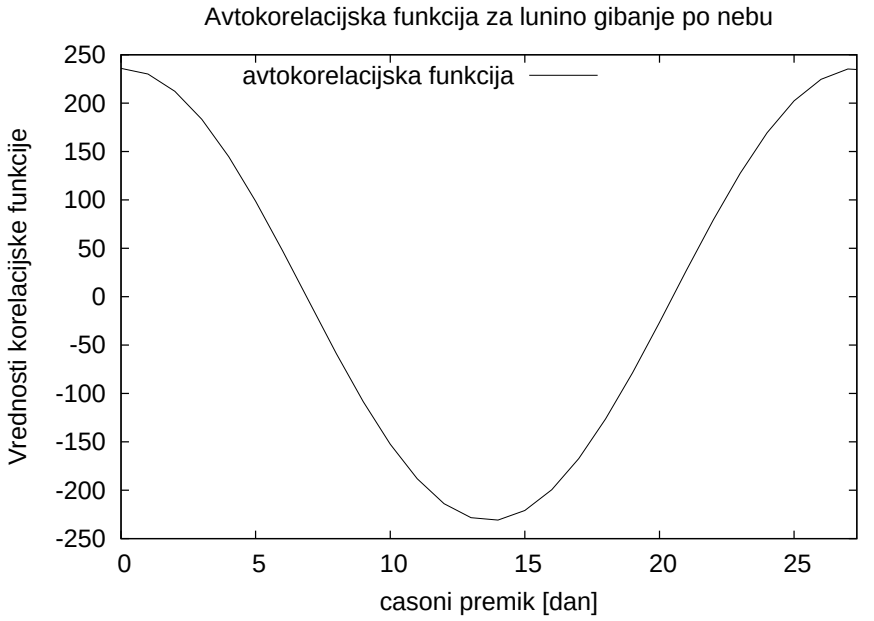
Avtokorelacijska funkcija izgleda takole:



Slika 4: Avtokorelacijska funkcija za gibanje lune.

Z grafa vidimo, da je funkcija maksimalna če ni premaknjena kar seveda očitno tudi iz definicije avtokorelacije.

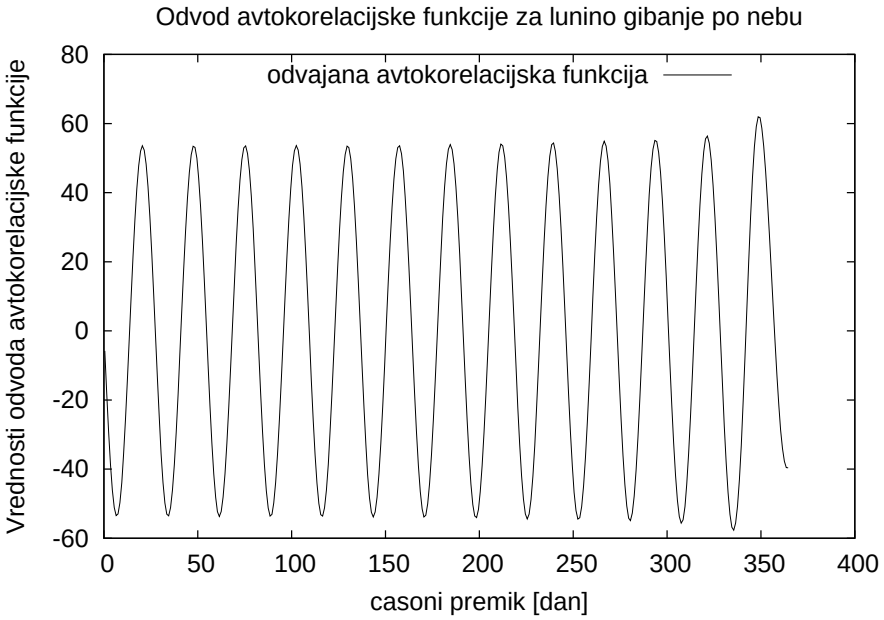
Povečajmo graf in odčitajmo eno periodo!



Slika 5: Avtokorelacijska funkcija za gibanje lune (interval ene same periode).

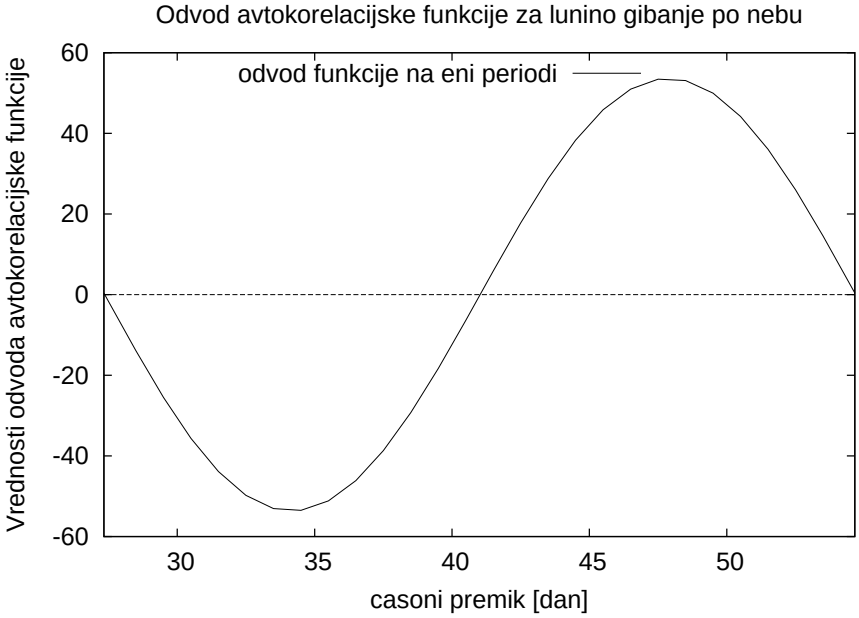
Iz grafa in izračunanih podatkov je očitno razvidno, da je en lunin mesec dolg 27 dni. Če dobro pogledamo graf vidimo, da je lunin mesec dolg v resnici malo več - 27.32 dni.

Graf, ki sem ga dobil bom še odvajal, da maksimini postanejo ničle in jeh bom lažje razbral:



Slika 6: Odvod avtokorelacijske funkcije za gibanje lune.

In še povečano:



Slika 7: Odvod avtokorelacijske funkcije za gibanje lune (interval ene same periode).

Iz grafa vidimo, da je Lunina period dolga natanko 27.32 dni.

Komentar: odvod je uporabljen samo zato, da ekstreme spremeni v ničle (iz sinusa naredi kosinus ali obratno)!