Računalniška orodja v fiziki

Skalarni produkt in korelacija

Avtor: Žiga Zaplotnik (vp. št.: 28031261)
Srednja Bela, 19. 4. 2009

1. naloga: določi korelacijski koeficient zveze med frekvenco rotorja in hitrostjo toka (vhodna datoteka: »*HitrostTokaOdFrekvence.txt*«)

Korelacijski koeficient zveze med obema količinama je

R(frekvenca, hitrost) = 0.987970

Komentar: korelacijski koeficient sem najprej izračunal v programu *Mathematica 7*.0 s funkcijo `Correlation(frekvenca, hitrost)`. Nato sem izračunal korelacijski koeficient še z lastnim programom, narejenim v programu *C* po spodnji formuli.

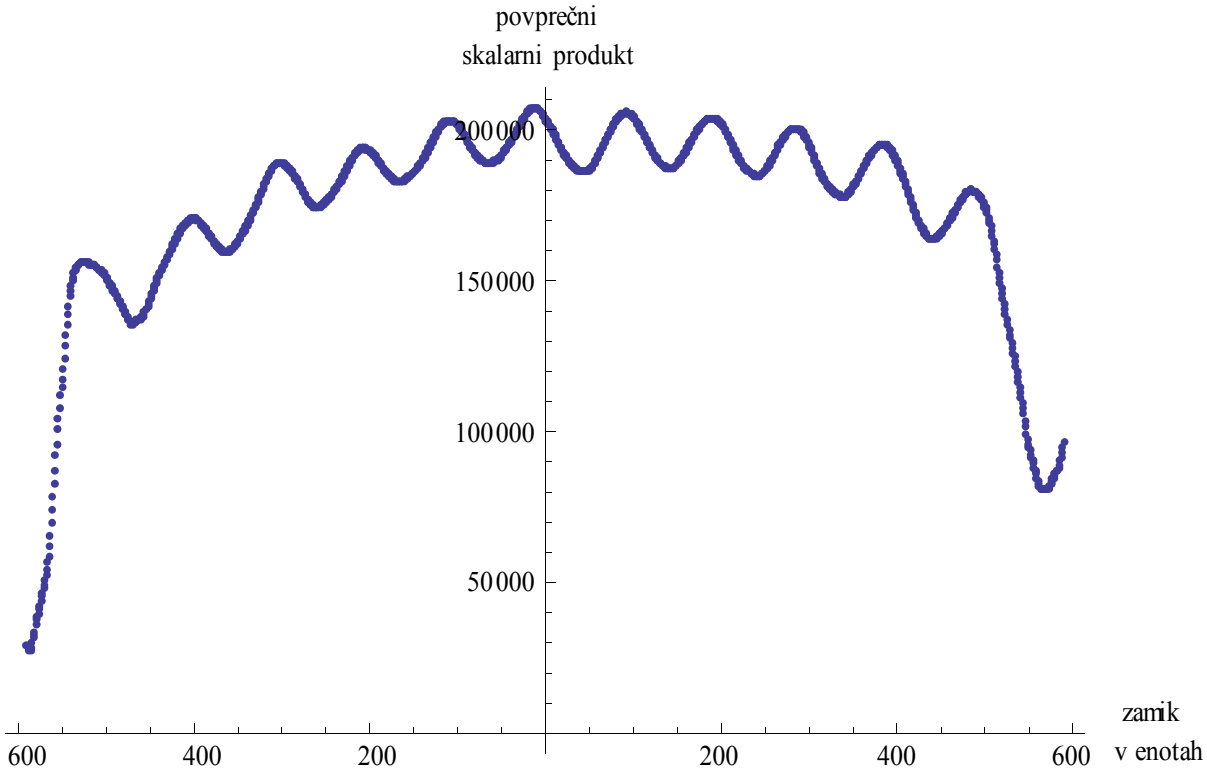
```
double skalab=0,r,R,sumfrek=0,sumv=0,sumfrek2=0,sumv2=0;

double sigmafrek,sigmav,frekpov,vpov;

for(int i=0; i<9; i++){

   skalab=skalab+frek[i]*v[i];

   sumv=sumv+v[i];

   sumfrek=sumfrek+frek[i];

}

frekpov=sumfrek/9;

vpov=sumv/9;

for(int i=0; i<9; i++){

   sumfrek2=sumfrek2+pow((frek[i]-frekpov),2);

   sumv2=sumv2+pow((v[i]-vpov),2);

}

sigmafrek=sqrt(sumfrek2/9);

sigmav=sqrt(sumv2/9);


r=skalab/9;

R=(r-frekpov*vpov)/(sigmafrek*sigmav);
```

2. naloga: določi korelacijski koeficient med dozo (v mg/kg žive mase) in stanjem bolezni po terapiji (ur zvonjenja na teden).(vhodna datoteka »Tintin.dat«)

Korelacijski koeficient zveze med obema količinama je

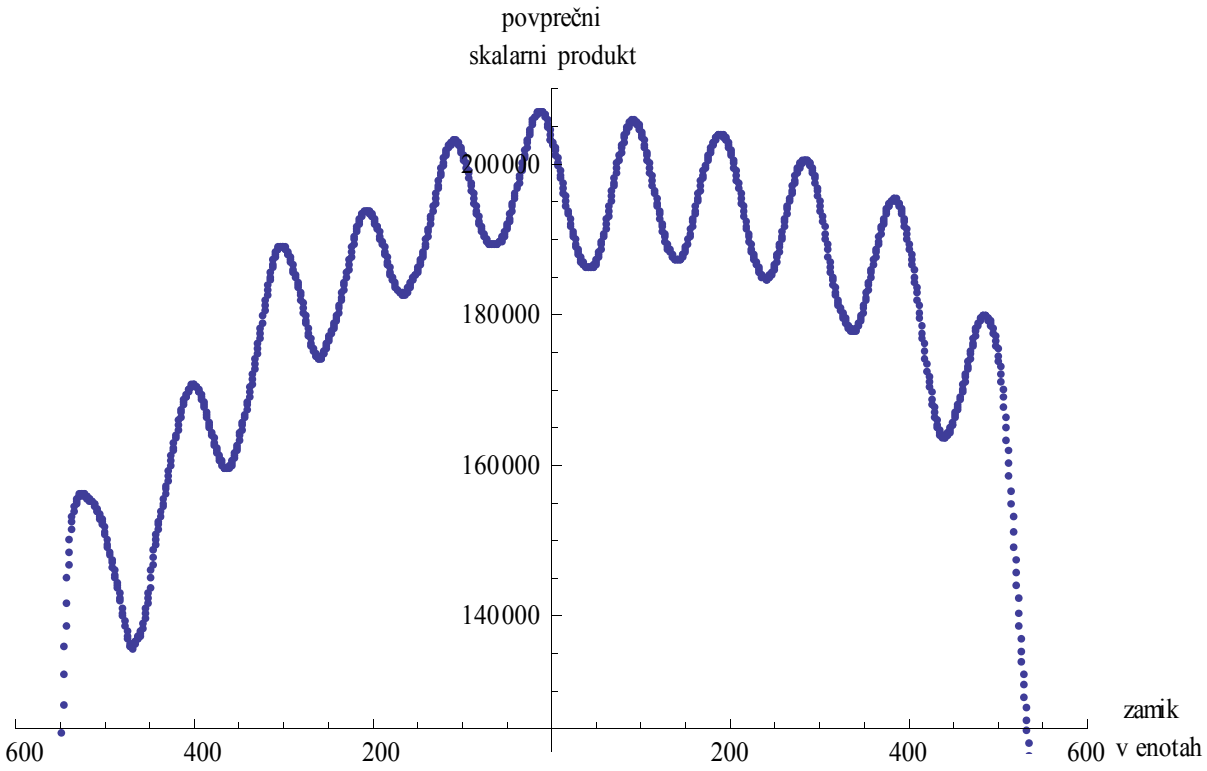
R(doza, stanje) = -0.39409

3. naloga: Določi efektivno zakasnitev med obema signaloma (meritvi temperature v bližini površine in globoko v notranjosti) iz njune korelacijske funkcije.



Komentar: zgornji graf prikazuje korelacijsko funkcijo med temperaturo blizu površine in temperaturo globoko v notranjosti betonskega bloka, spodnji graf pa je le povečan zgornji. Grafa sem narisal v programu *Mathematica 7.0*, podatke za graf pa sem predelal z lastnim programom, narejenim v programskem jeziku *C*. Spodaj je še izpisek glavnega dela programa.

```
int enota[MAX];
double t1[MAX], t2[MAX];
for(int i=591; i<1182; i++){
   int n=fscanf(vh, "%d""%lf""%lf", &enota[i], &t1[i], &t2[i]);
}
for(int i=1182; i<1773; i++){
   t1[i]=0;
   t2[i]=0;
}
for(int i=0; i<591; i++){
   t1[i]=0;
   t2[i]=0;
}
double skalab1[MAX], skalab2[MAX];
for(int y=0; y<591; y++){
   skalab1[y]=0;
   skalab2[y]=0;
}
for(int y=0; y<591; y++){
   for(int i=591; i<1182; i++){
      skalab1[y]=skalab1[y]+t1[i]*t2[i-y];
      skalab2[y]=skalab2[y]+t1[i]*t2[i+y];
   }
   skalab1[y]=skalab1[y]/(591-y);
   skalab2[y]=skalab2[y]/(591-y);
   fprintf(iz1,"%d        %lf\n", -y, skalab1[y]);
   fprintf(iz2,"%d        %lf\n", y, skalab2[y]);
}
```

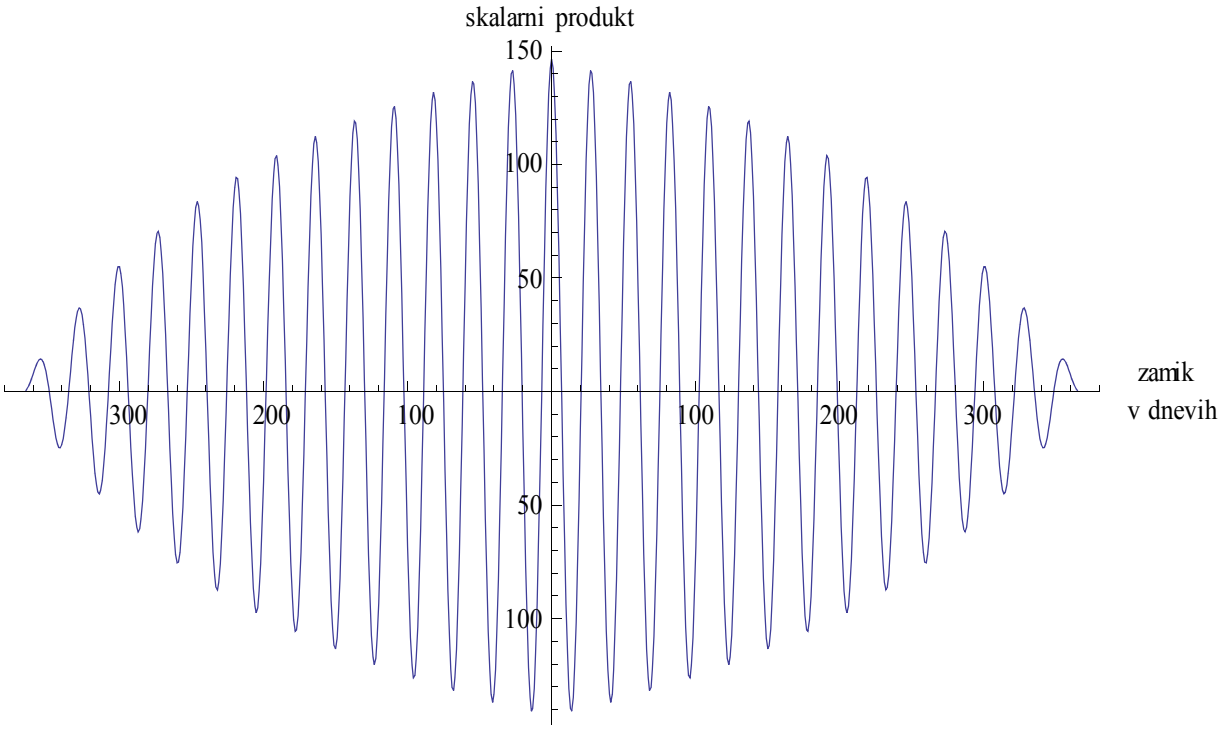


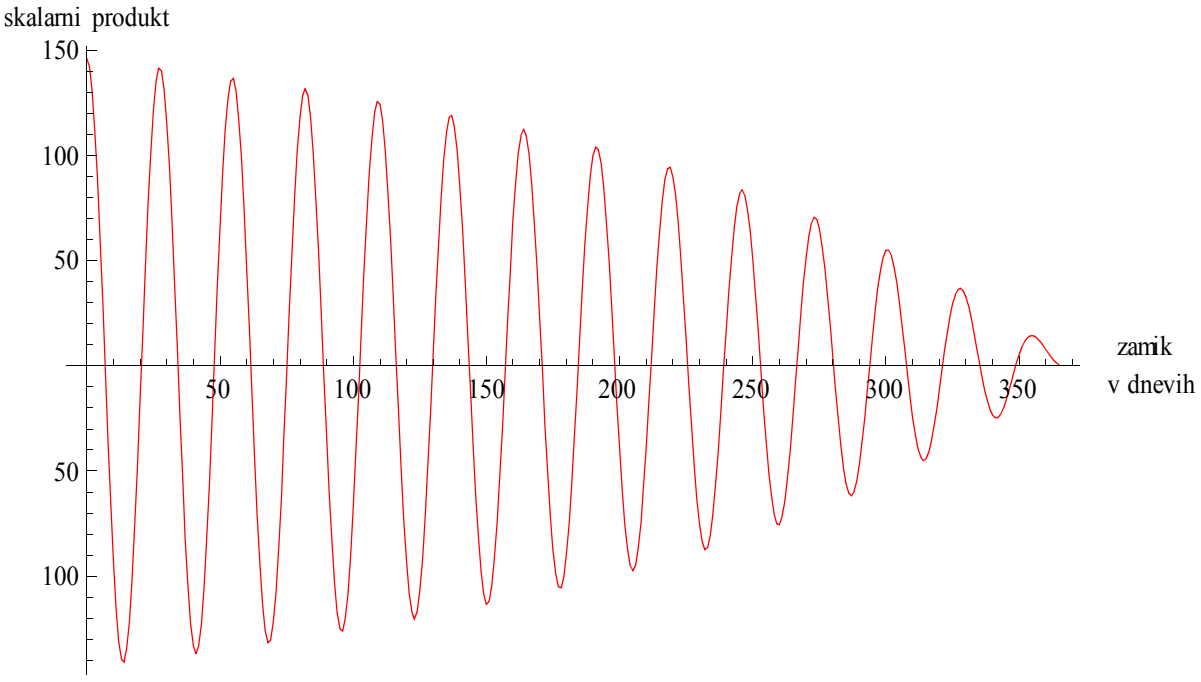
Iz grafov oz. korelacijske funkcije lahko razberemo efektivno zakasnitev med obema signaloma. Največji povprečni skalarni produkt (torej produkt glede na št. meritev) ni pri vrednosti 0 na grafu, temveč je pri premiku y = -10. (to sem ugotovil s pomočjo pregledovanja podatkov, nato pa še s pomočjo funkcije `GetCoordinate` v *Mathematici 7.0*.

Efektivna zakasnitev je torej $10 \cdot 14.619\,\text{min} = 146.19\,\text{min} = 146\,\text{min}$

Opomba: efektivna zakasnitev ni popolnoma točno določena, temveč je le ocena, predvsem zaradi spreminjanja dolžine period.

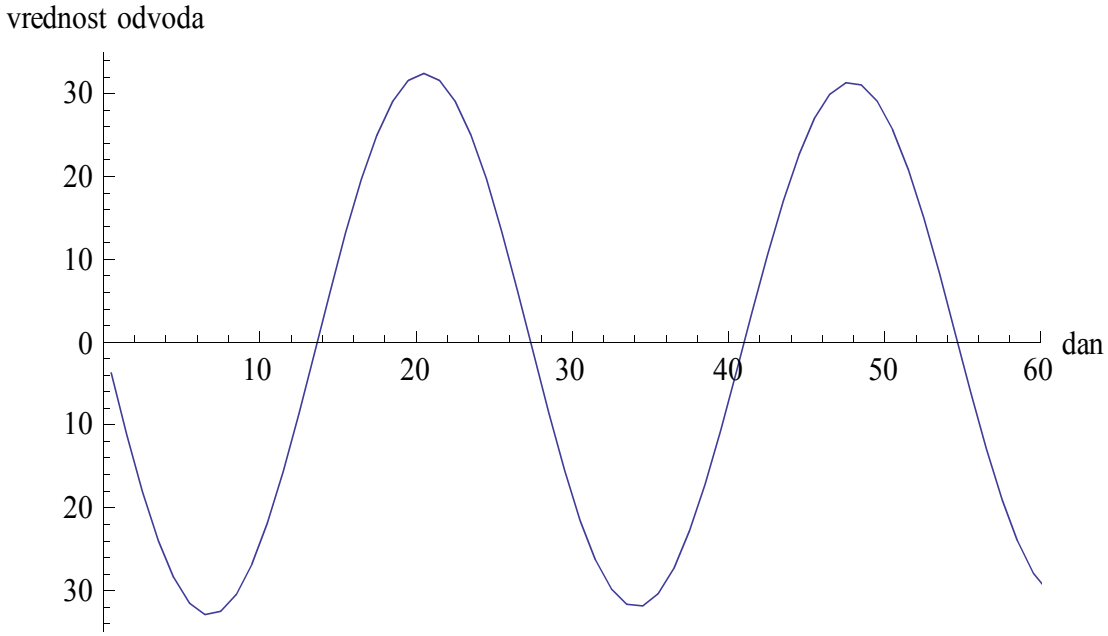
4. naloga: Iz avtokorelacijske funkcije deklinacije čim bolj natančno določi Lunino periodo tira. (vhodna datoteka: »Luna.efe«)





Komentar: grafa zgoraj prikazujeta avtokorelacijsko funkcijo za deklinacijo Lune. Narejena sta v programu *Mathematica 7.0*, podatki so prirejeni z lastnim programom, narejenim v program. jeziku *C*. Pri drugem grafu je definicijsko območje zoženo na pozitivne vrednosti.

Komentar: Lunina perioda tira je razdalja med dvema vrhovoma oz. dolinama avtokorelacijske funkcije. Razdaljo lahko določimo direktno iz podatkov ali pa z odvajanjem, pri čemer uporabimo formulo za diferenčni približek iz prejšnje naloge, narišemo funkcijo in pogledamo, kje ta funkcija seka os x oz. je vrednost odvoda enaka 0.



Rezultat (Lunina perioda tira) je 27.37 B 27.4 **dni.**

Spodaj sta še izpiska glavnih delov programov za računanje avtokorelacijske funkcije(zgornji) in odvoda (spodnji).

```
for(int j=0; j<366; j++){
   for(int i=367; i<733; i++){
      skalab1[j]=skalab1[j]+dek[i]*dek[i-j];
      skalab2[j]=skalab2[j]+dek[i]*dek[i+j];
   }
   skalab1[j]=skalab1[j]/(591-j);
   skalab2[j]=skalab2[j]/(591-j);}
}
```

```
for(int i=0; i<366; i++){
   odv[i]=((sk[i+1]-sk[i])/(dan[i+1]-dan[i]));
   dd[i]=(dan[i+1]+dan[i])/2;
```