6. Domača naloga - Skalarni produkt in koleracija

21. 4. 2009, Urban Škudnik

Hitrost toka od frekence

Pri prvem delu domače naloge smo določali korelacijski koeficient zveze med frekvenco rotorja hitrostjo toka, za katera dva so avtorji v svojem članku zatrdili, da sta močno linearno povezana. To smo uspeli z visokim koleracijskim količnikom tudi pokazati.

```
In[3]:= hittoka = ReadList[
          "/Users/urbanskudnik/Dev/racunalniska-orodja/skalpr/HitrostTokaOdFrekvence2.txt",
          {Real, Real, Real}]

In[6]:= frekvenca = First /@ hittoka

Out[6]= {4.42, 4.83, 4.93, 5.15, 5.28, 5.46, 5.62, 5.75, 6.71}

In[9]:= tok = hittoka[[All, 2]]

Out[9]= {2.55, 2.78, 2.81, 3.01, 3.26, 3.3, 3.48, 3.69, 4.8}

In[10]:= Correlation[frekvenca, tok]

Out[10]= 0.98797
```

Kot rečeno nam visok koleracijski količnik potrdi, da sta količini močno linearno odvisni, odstopanja pa lahko morda pripišemo napakam na opremi sami.

Čudežno zdravilo

V drugem delu smo si ogledali učinkovitost zdravila za zvonenje v ušesih, ki so ga opravili v ZDA pri ameriški upravi za prehrano in zdravila.

```
In[14]:= tintin = ReadList[
           "/Users/urbanskudnik/Dev/racunalniska-orodja/skalpr/Tintin2.dat", {Real, Real, Real}]

In[15]:= Correlation[tintin[[All, 2]], tintin[[All, 3]]]

Out[15]= -0.39409
```

Tokrat je korelacijski količnik mnogo manjši kot pri prvi nalogi, kar nam daje vedeti da je korelacija med količino prejetega zdravila in številom ur zvonenja v ušesih komajda zaznavna, zdravila pa ne moremo poimenovati čudežno zdravilo proti zvonenju v ušesih.

Beton

V tretjem delu se nismo več osredotočali samo na korelacijski koeficient, pač pa na korelacijsko funkcijo med temperaturo senzorja globoko v betonu in na njegovi površini..

In[16]:=
```
beton = ReadList[
  "/Users/urbanskudnik/Dev/racunalniska-orodja/skalpr/Beton.dat", {Real, Real, Real}]
```

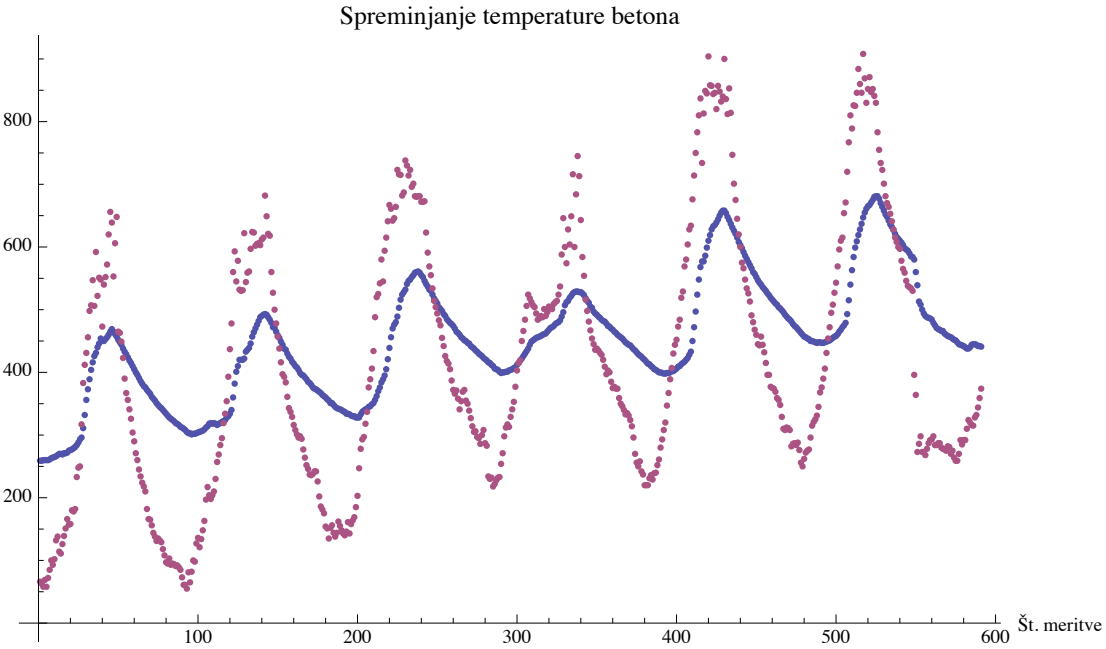
Meritve različnih senzorjev najprej razdelimo na dva vektorja.

In[26]:=
```
s1 = beton[[All, 2]]
```

In[27]:=
```
s2 = beton[[All, 3]]
```

In[155]:=
```
ListPlot[{s1, s2},
 PlotLabel → "Spreminjanje temperature betona", AxesLabel → {"Št. meritve"}]
```

Out[155]=



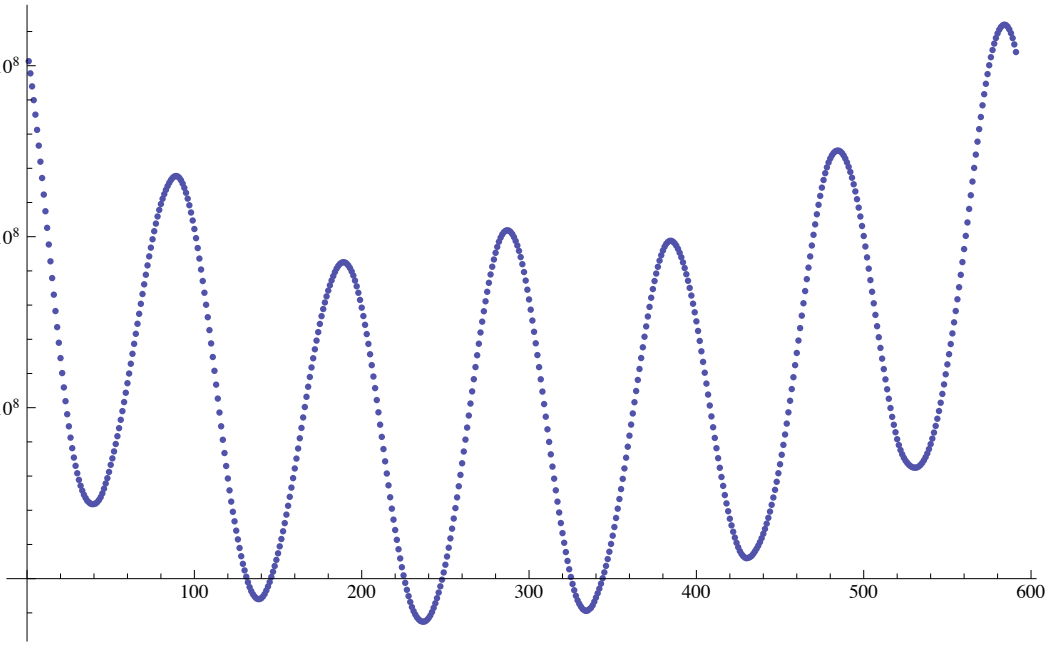
Za lažjo ugotovitev časovnega intervala si ogledamo kako se vrednosti spreminjajo s časom in, ker je meritev potekala ravno šest dni, je očitno, da je vsak dan bilo zajetih približno sto meritev oziroma nekaj več kot štiri na uro.

In[120]:=
```
ListCorrelate[s1, s2]
```

Out[120]= $\{1.20132 \times 10^8\}$

In[119]:=
```
ListPlot[ListCorrelate[s1, s2, 1]]
```

Out[119]=



Prvo število nam predstavlja korelacijski koeficient za naš niz podatkov, graf pa kako se le-ta spreminja s časom - vidimo lahko da je korelacija zelo šibka npr. pri stoti meritvi, ko je temperatura zunanjega betona močno padla, temperatura notranjega pa še ni imela časa pasti, ko pa se temperatura ponovno poveča razlika ponovno upade.

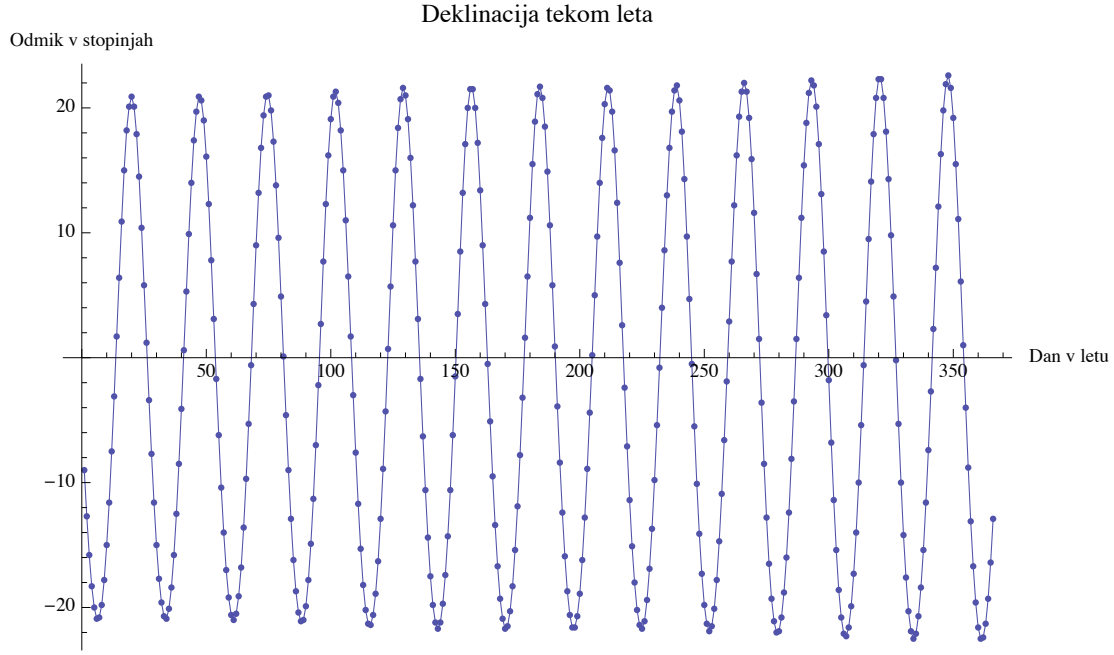
Luna

Pri četrti nalogi smo imeli opravka z krožejem lune okrog Zemlje v preteklem letu, pri čemer smo morali iz avtokorelacijske funkcije deklinacije izračunati Lunino periodo tira.

In[47]:=
```
luna = ReadList["/Users/urbanskudnik/Dev/racunalniska-orodja/skalpr/Luna.efe.txt",
  {Real, Real, Real, Real}]
```

In[152]:=
```
Show[ListLinePlot[luna[[All, 4]]], ListPlot[luna[[All, 4]]],
 PlotLabel → "Deklinacija tekom leta", AxesLabel → {"Dan v letu", "Odmik v stopinjah"}]
```

Out[152]=



In[51]:=
```
dek = luna[[All, 4]]
```

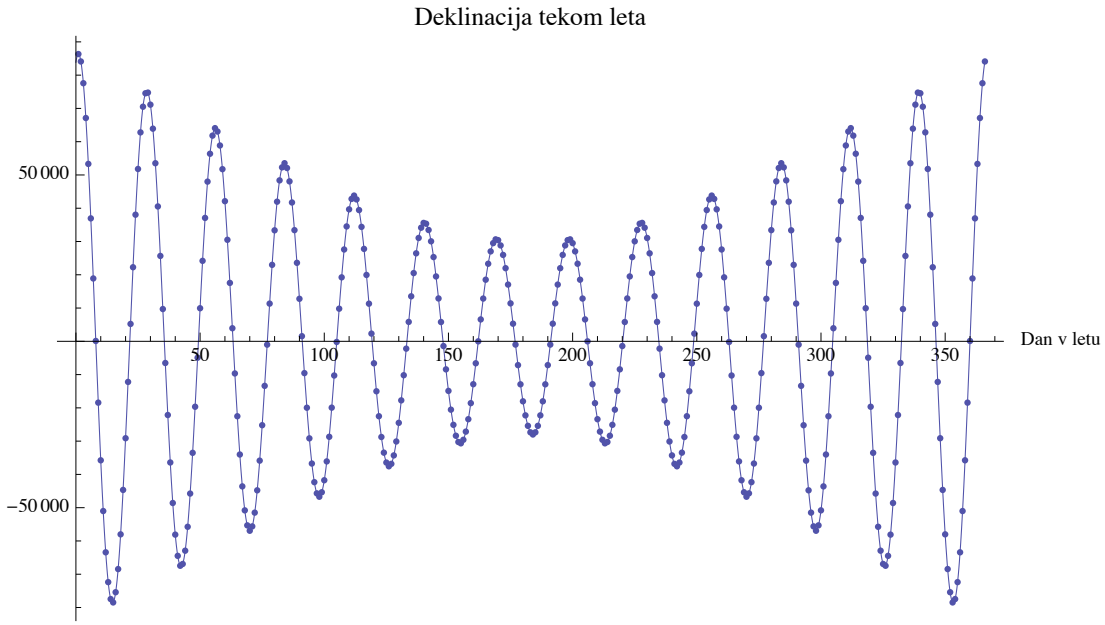
In[111]:=
```
co = ListCorrelate[dek, dek, 1]
```

In[86]:=
```
lu = Interpolation[ListCorrelate[dek, dek, 1]]
```

Out[86]=
```
InterpolatingFunction[{{1., 366.}}, <>]
```

In[154]:=
```
Show[ListPlot[co], Plot[lu[x], {x, 0, 366}],
 PlotLabel → "Deklinacija tekom leta", AxesLabel → {"Dan v letu"}]
```

Out[154]=



Kako je Mathematici uspela interpolacija funkcije preverimo z vrisanjem interpolirane funkcije in samih podatkov na en graf - vidimo lahko, da lahko s takšno interpolacijo lepo napredujemo in lahko izračunamo minimume (v našem primeru med mesecoma februarjem in julijem) ter razdalje med njimi.

In[132]:=
```
fmin[m_] := FindMinimum[{lu[x], m * 30 ≤ x ≤ (m + 1) * 30}, {x, m * 30}]
```

In[134]:=
```
minimumi = fmin /@ {1, 2 , 3, 4, 5, 6}
```

Out[134]=
```
{{-67 663., {x → 42.3441}}, {-56 933.8, {x → 70.047}}, {-46 714.5, {x → 97.9306}},
 {-37 570.5, {x → 126.099}}, {-30 697.5, {x → 154.776}}, {-28 003., {x → 184.}}}
```

Iz dobljenih podatkov izračunamo povprečje periode (zaradi "predvidevanja" rezultata v programu Mathematica podatke najprej pretvorimo v realna števila).

In[138]:=
```
mine = {42.34405758424798, 70.04704026013114, 97.93064746027584,
  126.09880643104587, 154.77565053054474, 184.00000000926883}
```

Out[138]=
```
{42.3441, 70.047, 97.9306, 126.099, 154.776, 184.}
```

In[148]:=
```
povf[i_] := (mine[[i + 1]] - mine[[i]])
```

In[150]:=
```
povf /@ {1, 2, 3, 4, 5}
```

Out[150]=
```
{27.703, 27.8836, 28.1682, 28.6768, 29.2243}
```

Iz sledečih period lahko sedaj končno izračunamo tudi povprečno periodo (v številu dni).

In[151]:=
```
Mean[%]
```

Out[151]=
```
28.3312
```