Univerzitet u Banjoj Luci
Elektrotehnički fakultet
Katedra za opštu elektrotehniku
Osnovi digitalne obrade signala

Domaći zadatak 2

Autor: Saša Pejašinović

O softveru

Uputsvo za upotrebu

Program je implementiran u okviru radar.ipynb fajla. Ćelije sa kodom nisu nezavisne jedna od druge, pa ih je neophodno izvršavati jednu po jednu, od početka prema kraju fajla. Program je implementiran korak po korak vodeći se redom kojim su raspoređeni zadaci unutar fajla radar_zadatak.pdf.

Program je napisan tako da važi za sve zadane radar_nnn.mat fajlove, a i druge koji poštuju ovakvu formu.

Ispod sekcije 13. u okviru fajla radar.ipynb nalazi se celija sa sa sledećim kodom.

In [ ]:
```
import scipy.io
import math
d=scipy.io.loadmat("radar_nnn.mat")
```

Da bi smo testirali različite signale koji su definisani u .mat fajlovima, dovoljno je samo da navedemo ime fajla kao argument funkcije scipy.io.loadmat("radar_nnn.mat") i pokrenemo program počevši od te ćelije.

Obrazloženja izbora pojedinih parametara algoritma

Program je napisan sa željom da se složenost programa, odnosno algoritama korištenih u implementaciji svede na minimum, ali i da kod bude jednostavan i čitljiv. Pa shodno tome postoje dijelovi programa u kojima je tražen kompromis između ova dva zahtjeva.

Izbor pojedinih funkcija, kao što je np.convolve i signal.fft.fftconvolve je testiran. Npr. kod propuštanja svakog impulsa u 13. zadatku kroz usklađeni filtar, pokazuje se da np.convolve zahtjeva kraće vrijeme izračunavanja od signal.fftconvolve.

In [26]:
```
%%timeit -n100
filtered=np.apply_along_axis(signal.fftconvolve,1,y,h,mode='same').T
```

```
2.83 ms ± 362 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [27]:
```
%%timeit -n100
filtered=np.apply_along_axis(np.convolve,1,y,h,mode='same').T
```

```
867 µs ± 137 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Međutim postoje i zadaci u kojima se funkcija signal.fftconvolve pokazala kao bolji izbor.
Takođe na slici se iznad, može se primjetiti da kao argument funkcije np.convolve uzimamo mode='same'. Ako je chirp superponiran u signal na ulazu u filtar, tada će se impuls koji se dobija na izlazu usklađenog filtra sigurno nalaziti unutar istog intervala u kojem se nalazi ulazni signal, iz tog razloga koristimo mode='same'. Pa prema tome nema potrebe da posmatramo izlazni signal van tog intervala, odnosno intervala u kojem se nalazi ulazni signal.

Prilikom korištenja funkcije funkcije np.fft.fft, radi postizanja boljih performansi vodio sam računa da dužina signala bude proširena nulama tako da konačna dužina signala bude stepen broja 2. Na primjer prilikom traženja dtft (npr. decimiranih signala zbog kratkog trajanja tj. duzine), radi postizanja bolje aproksimacije. Međutim ukoliko je signal dovoljne dužine tako da nema potrebe da se proširuje nulama, kao što je to slučaj kod diskretnog signala koji predstavlja povorku LMF impulsa, nisam radio dodatno proširivanje radi postizanja boljih performansi algoritma fft jer se pokazalo da je funkcija fft bez proširivanja nulama do dužine stepena broja 2 dala bolje rezultate. Dokaz za to prikazan je na sledecoj slici, gdje x predstavlja odaslani signal u 13. zadatku.

In [60]:
```
print(len(x))
n = np.int(2**(np.ceil(np.log2(len(x)))))
print(n)
```

```
9600
16384
```

In [61]:
```
%%timeit
X=np.fft.fft(x,n)
X=np.fft.fftshift(X)
```

```
523 µs ± 78.7 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [62]:
```
%%timeit
X=np.fft.fft(x)
X=np.fft.fftshift(X)
```

```
246 µs ± 2.15 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Na slici možemo primjetiti da smo manje vrijeme izračunavanja dobili bez proširivanja nulama do dužine stepena broja 2.

Funkcija diskretni_chirp implementirana je na sledeći način

In [2]:
```
import numpy as np
def diskretni_chirp(T, W, p):
    c=1j*np.pi*W*np.square(np.arange(-T/2,T/2,1/(W*p)))/T
    x=np.exp(c)
    return x
```

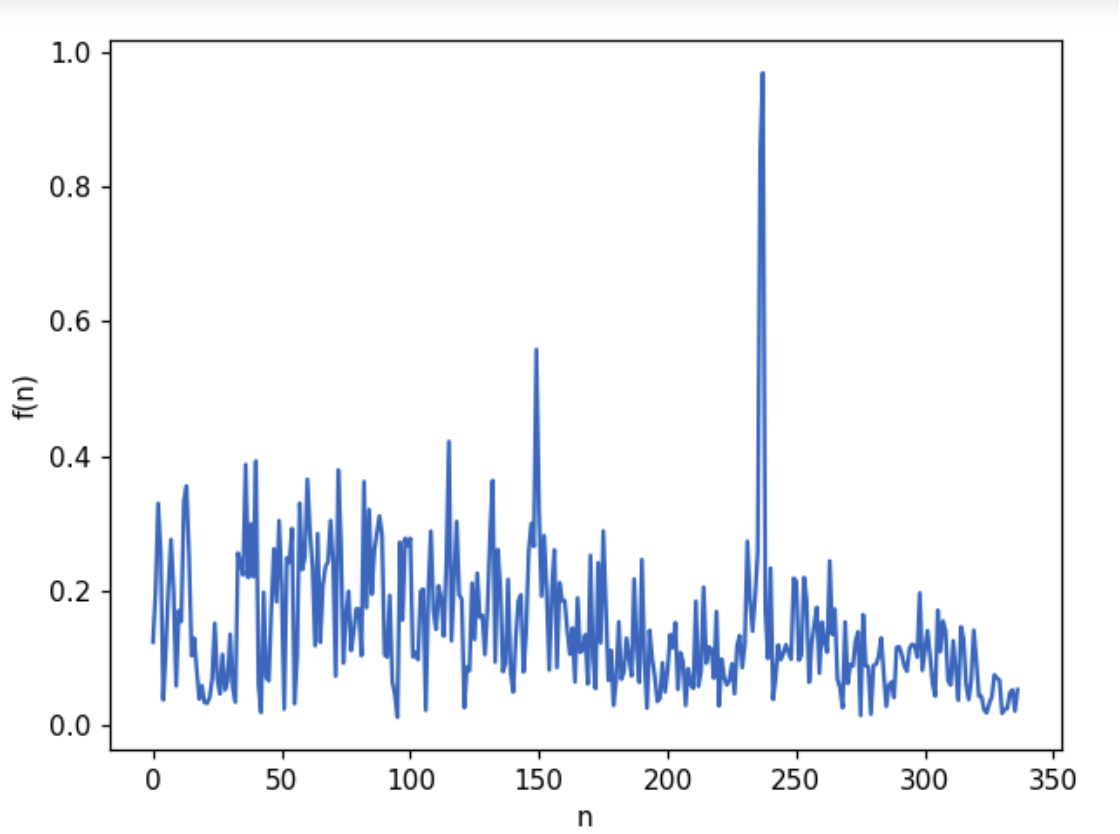
i pored što je iskorištena za generisanje niza koji predstavlja diskretizovan chirp signal, da bi se izbjeglo dupliranje koda, iskorištena je i za generisanje impulsnog odziva usklađenog filtra i to na sljedeći način. Kako se impulsni odziv uskladjenog filtra i chirp razlikuju samo za minus u eksponentu, moguće je impulsni odziv uskladjenog filtra predstaviti kao

In [ ]:
```
h=diskretni_chirp(T,W,p)**(-1)
```

Analiza dobrih strana i nedostataka

Kao što je već rečeno u prethodnoj sekciji, program je optimizovan u skladu sa mojim vještinama iz programskog jezika Python i digitalne obrade signala, tako da sigurno postoji prostor za dalju optimizaciju u pogledu performansi, kao i same obrade. Moguće je napraviti automatsku detekciju ciljeva. Nedostatak je to što je sa samog grafika teško i sa ograničenom preciznosti detektovati mete i izračunati tačne udaljenosti i brzine ciljeva.
Nedostatak je i nemogućnost detekcije velikih brzina (zbog maksimalnog Doplerovog pomaka), kao i ograničena rezolucija.

Detekcija ciljeva

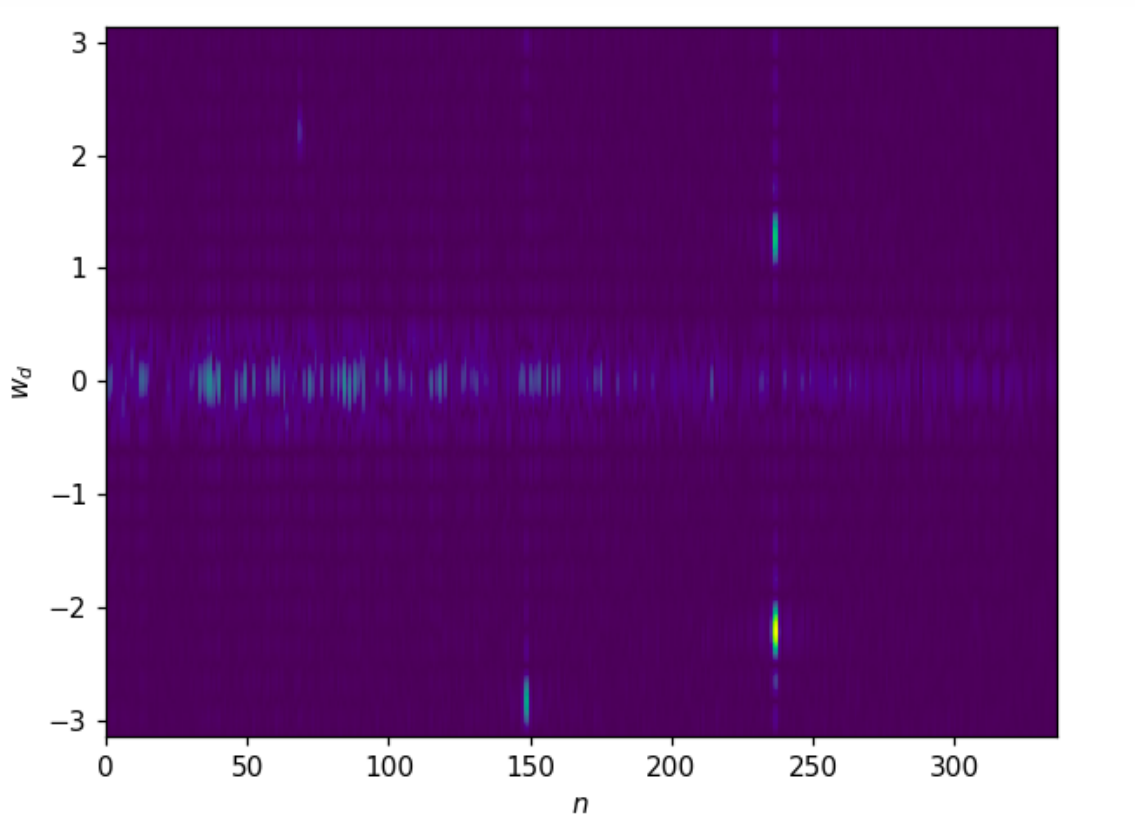


Na grafiku iznad prikazan je odziv usklađenog filtra za prvi impuls iz primljenog radarskog signala. Odnosno prve kolonu matrice y iz fajla radar_008.mat. Napomena, moj radarski signal je radar_003.mat, ali sam u dokumentaciji koristio 008 zbog demonstracije i lakšeg uočavanja ciljeva.

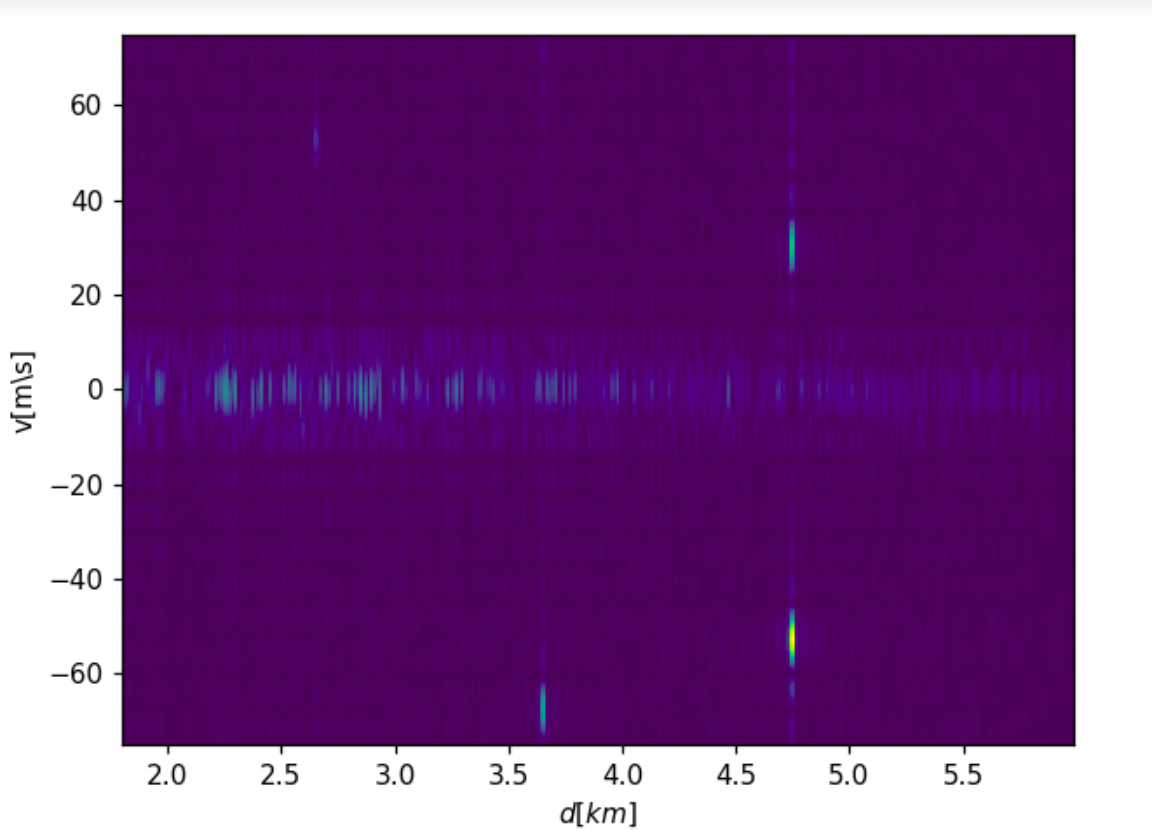
Na grafiku se jasno vidi da nemamo samo četri uska impulsa velikih ampliutuda koja bi predstavljala 4 različita cilja. Pored impulsa ciljeva (koje trenutno ne možemo uočiti), postoje i impulsi nastali tako što je radarski signal odbijen od nepokretne objekte ili objekte malih brzina kretanja, odnosno irelevantni signali.

Da bismo uočili pokretne ciljeve, potrebno je da odredimo Doplerov pomak tako što ćemo za svaki odmjerak iz prvog impulsa uzimati odmjerak po odmjerak na istoj poziciji iz svakog sledeceg impulsa (13. zadatak,tačka broj 2). Tako ćemo dobiti N signala, gdje je N broj odmjeraka u prvom impulsu. Nađemo amplitudski spektar svakog od signala, i iz amplitudskog spektra odredimo Doplerov pomak za svaki cilj prikazan na grafiku iznad. Samo oni ciljevi koji imaju Doplerov pomak različit od nula odnose se na pokretne ciljeve.

Ukratko, za svaki odmjerak iz prvog primljenog impulsa, imaćemo spektar iz kojeg možemo odrediti Doplerov pomak, odnosno brzinu, a iz pozicije odmjerka u prvom impulsu i zadanih parametara možemo odrediti udaljenost cilja. Da ne bismo posmatrali svaki odmjerak i svaki spektar posebno, moguće je ovo prikazati na sledećem spektogramu.



Doplerov pomak određuje se na osnovu pozicije glavnog luka u spektru decimiranog signala, više o ovom u fajlu radar.ipynb. Na spektogramu veće vrijednosti obojene su jarkim bojama, samim tim lako je uočiti glavni luk svakog od spektara. S obzirom na to da pozicija glavnog luka u spektru određuje Doplerov pomak, lako je uočiti da na prikazanom spektru imamo samo četri glavna luka koja nisu u nuli, pa se oni odnose na pokretne ciljeve, svi ostali odnose se na nepokretne objekte i objekte malih brzina. Konverzijama osa prikazanog spektograma dolazimo do sljedećeg grafičkog prikaza, iz kojeg pozicija glavnog luka na isti način određuje brzinu i udaljenost cilja. Za određivanje brzine i udaljenosti dovoljno je da se kursorom pozicioniramo na neku od meta (na glavni luk) i očitamo pozicije u donjem lijevom uglu



S obzirom na to da nisam radio program u fajlovima sa .py ekstenzijom, odnosno nisam pisao samo kod. Svi grafici traženi u zadacima nacrtani su fajlu sa kodom, odnosno u radar.ipynb fajlu, pa ih nisam crtao u dokumentaciji.