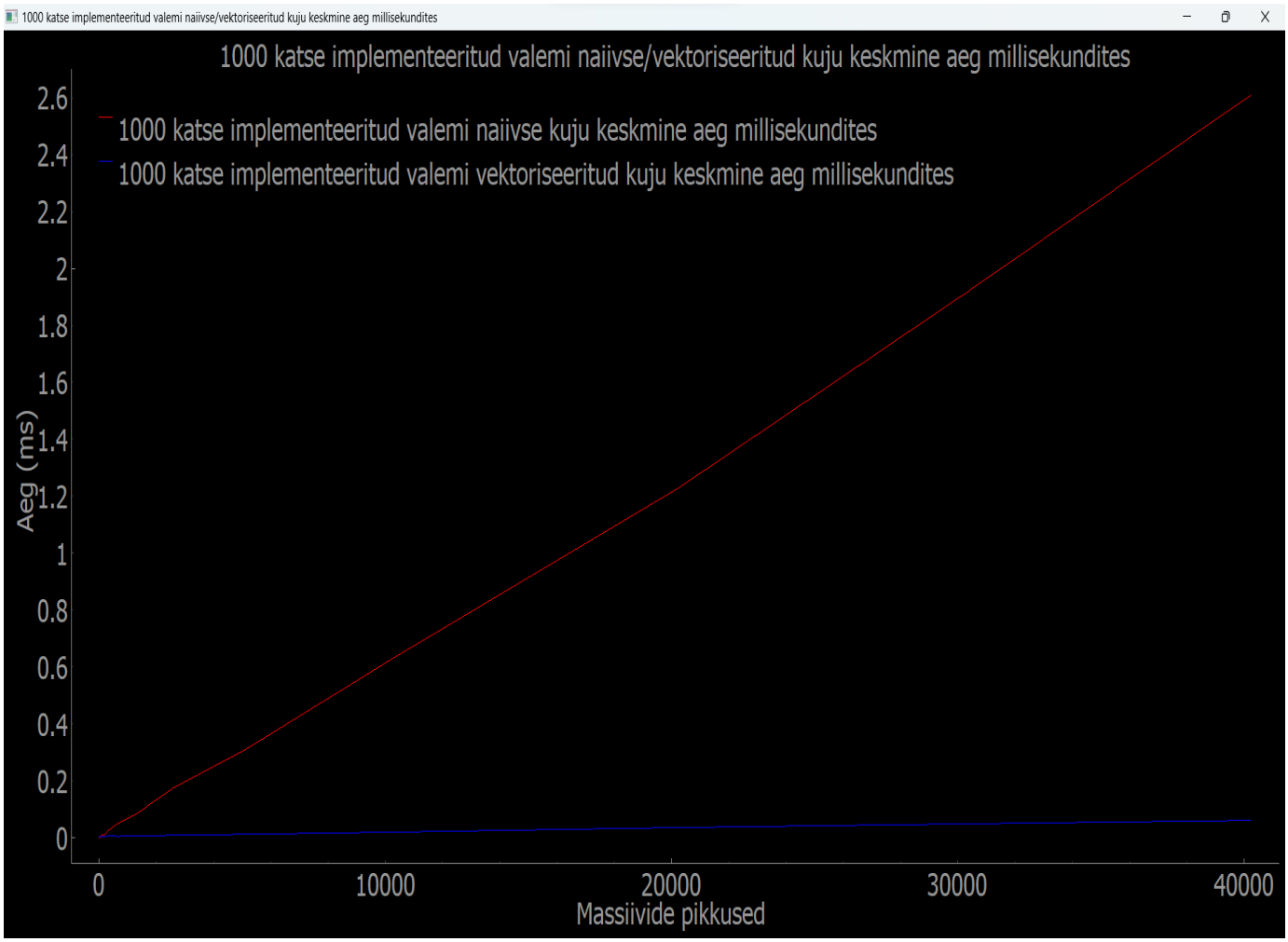
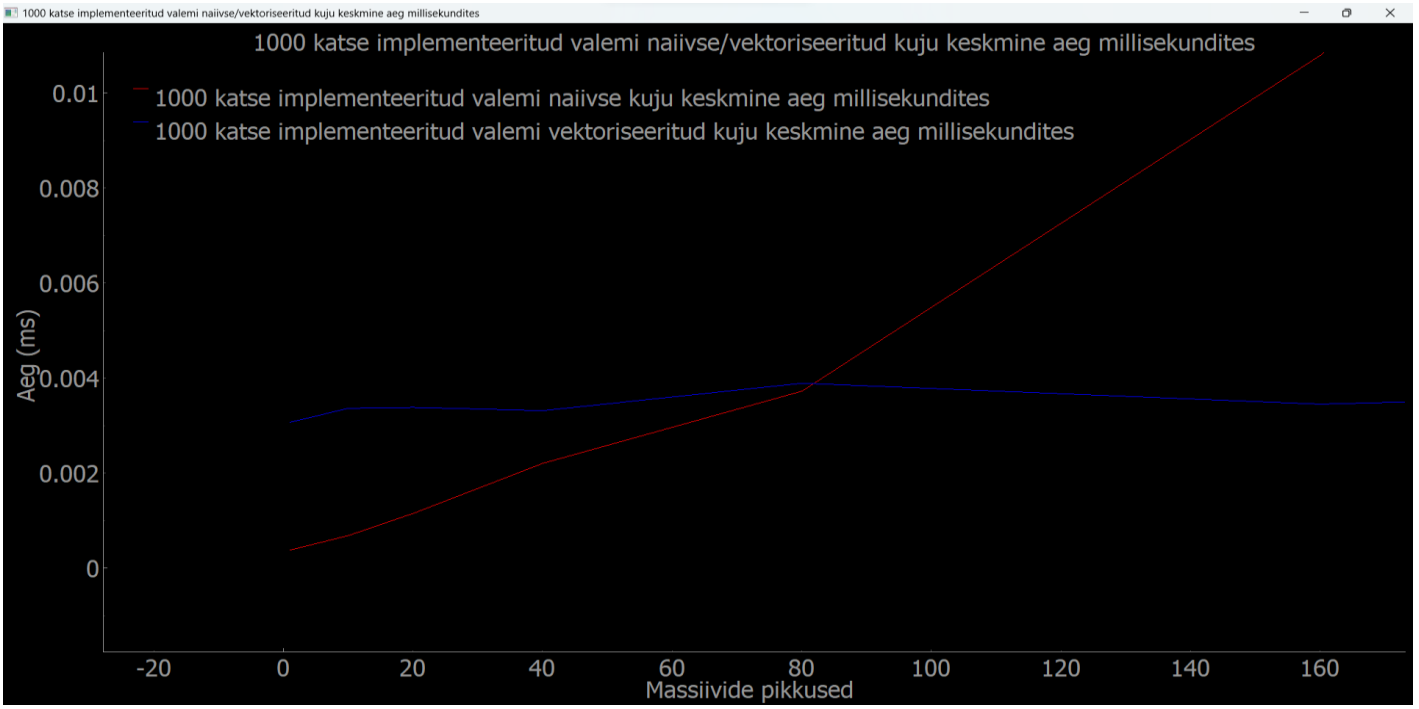
Kodutöö ülesanded

1.1 Ülesanne

Sisendite massiivide pikkused	1000 katse implementeeritud valemi naiivse kuju keskmine aeg millisekundites	1000 katse implementeeritud valemi vektoriseeritud kuju keskmine aeg millisekundites
1	0,0003396	0,00321459
10	0,00064849	0,00330979
20	0,001092	0,00334189
40	0,00197979	0,00346719
80	0,003671	0,00342199
160	0,00751269	0,0037796
320	0,01576239	0,0043431
640	0,03387189	0,00503239
1280	0,0729208	0,005724
2560	0,15240439	0,00803379
5120	0,276483	0,01197009
10240	0,56773079	0,0189484
20240	1,14806649	0,0320196
30240	2,0148322	0,0544436
40240	2,54254409	0,0585358
Aeg kokku millisekundites	6,83985961	0,21958582


1000 katse implementeeritud valemi naiivse/vektoriseeritud kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi naiivse/vektoriseeritud kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi naiivse kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi vektoriseeritud kuju keskmine aeg millisekundites
Aeg (ms)
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10000
20000
30000
40000
Massiivide pikkused


1000 katse implementeeritud valemi naiivse/vektoriseeritud kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi naiivse/vektoriseeritud kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi naiivse kuju keskmine aeg millisekundites
1000 katse implementeeritud valemi vektoriseeritud kuju keskmine aeg millisekundites
Aeg (ms)
0.01
0.008
0.006
0.004
0.002
0
-20
0
20
40
60
80
100
120
140
160
Massiivide pikkused

Vastus: Vektoriseeritud valemi(0,21958582 millisekundit) implementeeritud töökiirus on kiirem kui naiivse valemi(6,83985961 millisekundit) töökiirus. Vektoriseeritud valem kasutab matemaatiliste arvutuste tegemiseks vektoreid ja massiive, mis võimaldab kiiremini lahendada valemeid kui naiivse valemiga arvutamine, kus kasutatakse tsükleid ja tingimuslausetega arvutusi. Lisaks panin tähele, et lihtsamate (Sisendite massiivide pikkused 1 – 40) arvutuste puhul, kus iga iteratsiooni arvutuskoormus on minimaalne on implementeeritud valemi naiivse kuju töökiiruse aeg kiirem, kui implementeeritud valemi vektoriseeritud kuju töökiiruse aeg.

1.2 Ülesanne

numpy.correlate - NumPy v1.26 Manual

Ma arvan, et funktsioon, mis mind võiks tulevikus aidata on “numpy.correlate()”,

Selle funktsiooniga on minul võimalik arvutada signaalide ristkorrelatisooni. Ristkorrelatsiooni saame kasutada näiteks kahe helisignaali sarnasuse tuvastamisel.

Samamoodi on ka võimalik erinevate helisignaalide vahel tuvastada erinevaid seoseid. “numpy.correlate()” funktsioon on näiteks kasulik helitöötluse, biomeditsiini ja pilditöötluse valdkondades.

“numpy.correlate(a, v, mode=”valid”)” – selles funktsioonis on kaks sisendsignaali – a ja v. See funktsioon tagastab nende mõlema sisendsignaali(a,v) ristkorrelatsiooni. Mode=”valid” režiim annab väljundi, kui sisendsignaalid a ja v kattuvad täielikult.

```
Import numpy as np

# kaks signaali

a = [8, 1, 3, 6, 7]

v = [1, 2, 3]


# Arvutame ristkorrelatsiooni kahe signaali vahel

Ristkorrelatsioon = np.correlate(a, v, mode = “valid”)


# Arvutuskäik
```

Korrutame samadel positsioonidel olevad elemendid. Esimese elemendi väärtuse (a) korrutame signaaliga (v). Lõpuks liidame korrutised kokku.

8*1 + 1*2 + 3*3 = 19

Teise elemendi vääruse (a) korrutame signaaliga (v). Lõpuks liidame korrutised kokku.

1*1 + 3*2 + 6*3 = 25

Viimaseks kolmanda ehk viimase (a) elemendi väärtuse korrutame signaaliga (v). Lõpuks liidame korrutised kokku.

3*1 + 6*2 + 7*3 = 36

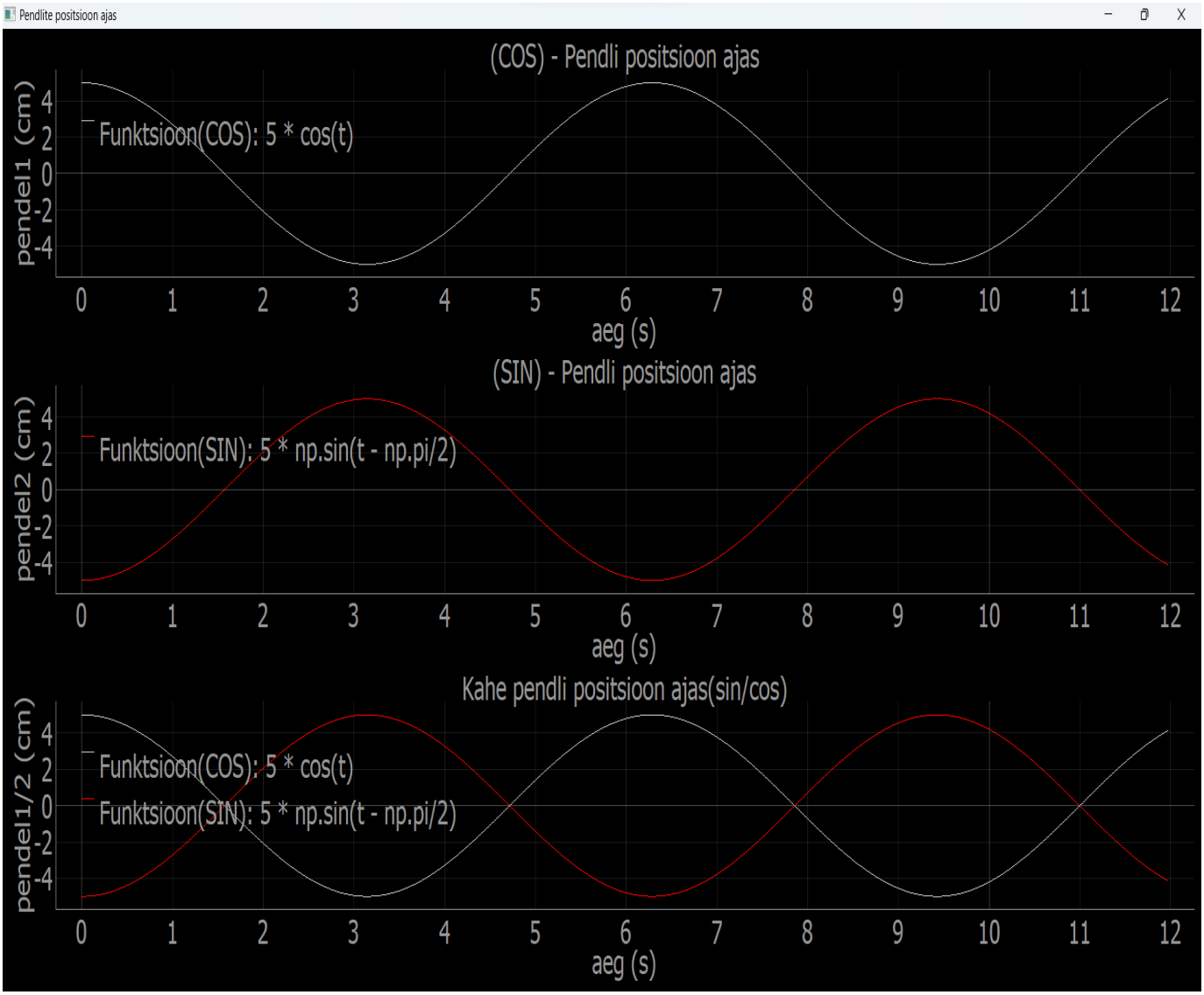
Saame vastuseks array([19, 25, 36])

Ristkorrelatsiooni käigus saame uue signaali, mis sisaldab ainult neid väärtuseid, kus signaalid kattuvad.

2.1 Ülesanne

Ringsagedus – Vikipeedia (wikipedia.org)

https://et.wikipedia.org/wiki/Lihtharmooniline_v%C3%B5nkumine



Graafikute siinus- ja koosinusfunktsioonid on lihtharmoonilise võnkumise funktsioonid:

$x(t)=A\sin(\omega t+\phi 0)$ või $x(t)=A\cos(\omega t+\phi 0)$

y = A*sin(Bt + C)

A on võnkeamplituud,

ω on ringsagedus

Ringsageduse valem:

$\omega=2\pi f=2\pi/T$

funktsiooni perioodi T = (2 * π) / ω,

$\phi 0$ on võnkumise algfaas,

t on aeg.

Antud juhul, kus t = np.arange(0, 12, 0.03) ja funktsioonid on defineeritud kui 5*cos(t) ja 5*sin(t - π /2), on ω = 1 mõlemal juhul. Mõlema funktsiooni periood on **(T = (2 * π) / ω)** ehk **(T = (2 * π) / 1) = 2 π sekundit**. Kuna t on antud sekundites, on ühe täisringi **360 kraadi ehk 2 π** sekundi läbimiseks vajalik aeg periood **T**.

Periood **T = 2 π** sekundit ehk ligikaudu **6.28** sekundit, mis tähendab, et funktsioon kordub iga **2 π** sekundi ehk **6.28** sekundi järel.

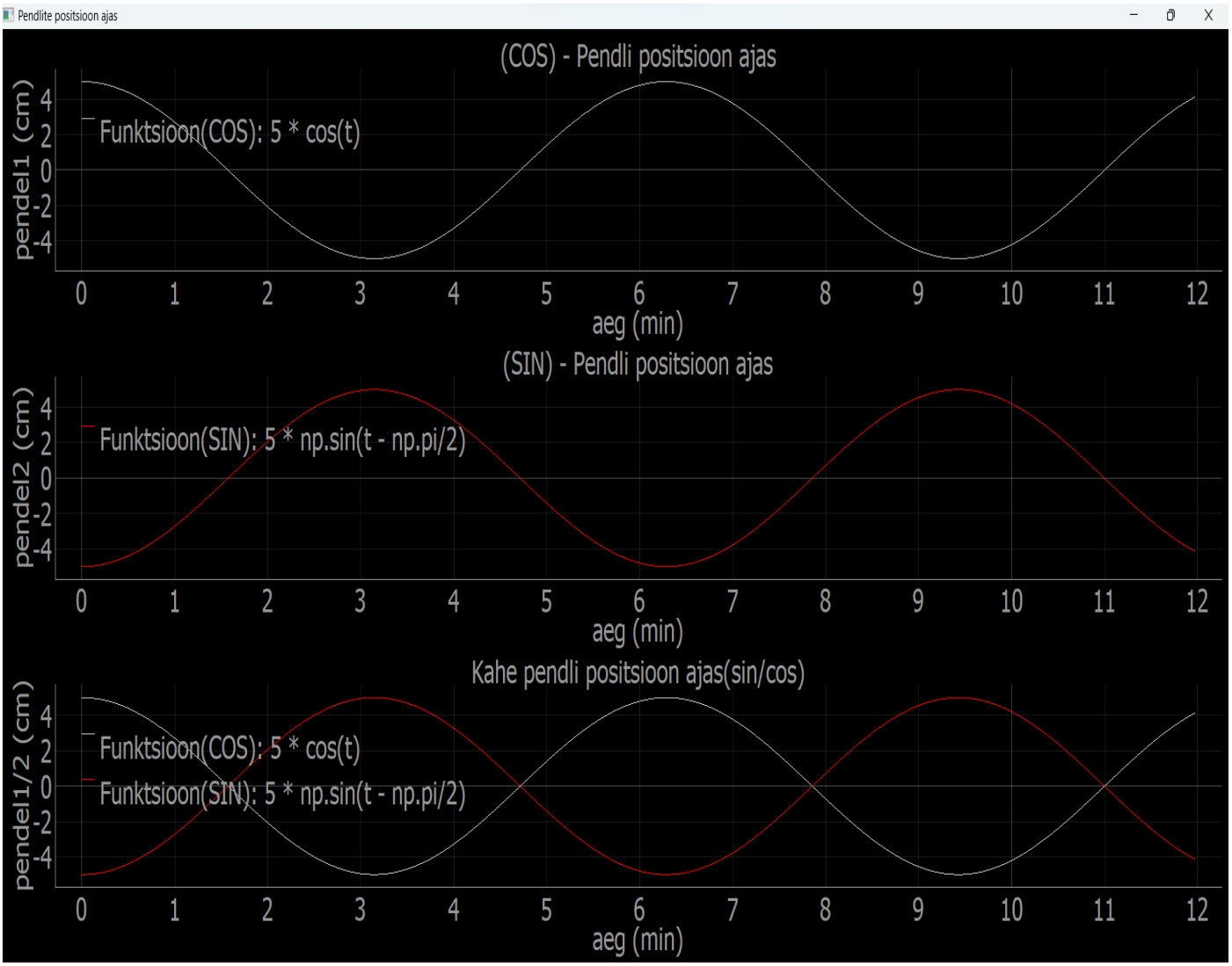
Sagedus **f = 1/T = 1/2 π Hz**, mis täehndab, et ühe sekundi jooksul toimub **1/2 π** täis tsüklit ehk **0.159 Hz**.

Kui X-telg Tähistab **Aega 0-st 12 Minutini**

Kui x-telje väärtused esindavad aega minutites **0-st 12-ni**, muutub ajaskaala, kuid funktsioonide põhivormid jäävad samaks. Siiski, et määrata perioodi ja sagedust selles kontekstis, peame teisendama ajaskaala sekunditeks, kuna standardühikud sageduse jaoks on hertsid (tsüklid sekundis).

12 minutit on 720 sekundit.

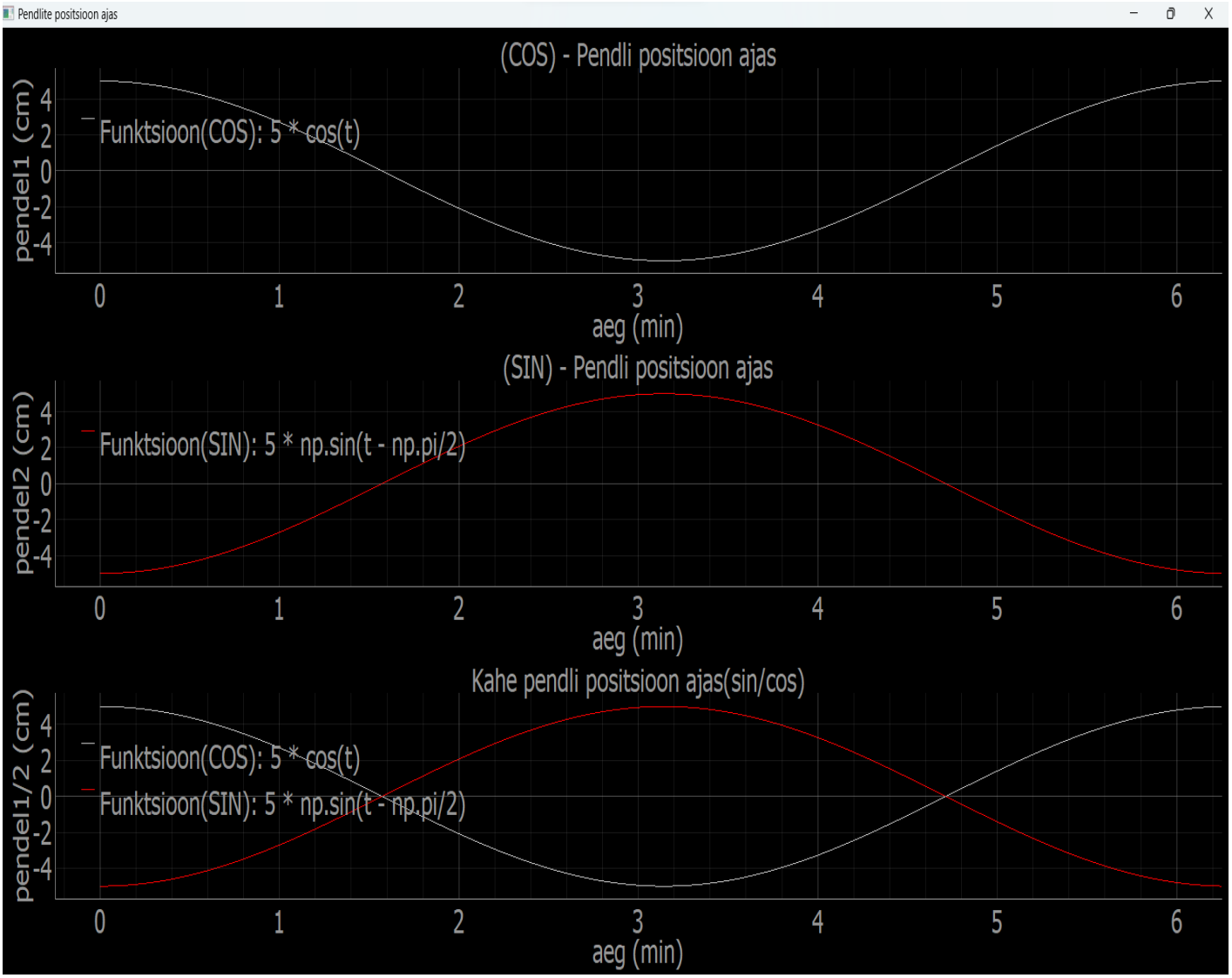
Kui eeldame, et t on nüüd sekundites, kuid meie funktsioonide definitsioonid jäävad samaks **(5*cos(t) ja 5*sin(t - π /2))**, siis periood on **2 π minutit** ehk ligikaudu **6.28 minutit ehk ligikaudu 377 sekudit**. See tähendab, et iga **2 π minuti** järel funktsioon kordub. Kuna meie ajaskaala on minutites, siis agedus f samuti ei muutu väärtuslikult, kuid tuleb mõista, et see on defineeritud tsüklitena minutites ehk **1/2 π minuti ehk 0.159 Hz** järel läbib funktsioon ühe täieliku tsükli. Sekundites teeks see **0.159 / 60 = 0.00265 Hz**



2.2 Ülesanne

https://pyqtgraph.readthedocs.io/en/latest/api_reference/graphicsItems/viewbox.html#pyqtgraph.ViewBox.setXRange

Valisin setXRange() funktsiooni PyQtGraph'is. See funktsioon võimaldab minul graafikul määrata x-telje vahemiku. See on minule kasulik, kui näiteks soovin ainult teatud ajavahemikku kuvada. Soovin ainult näiteks kuvada 6.28 minuti ajavahemikku ja uurida graafikute perioodi ja sagedust. Lisasin enda varasemalt kirjutatud 1. Kodutöö_v1 koodi yl2.2(): koodiread: p1.setXRange(0, 6.28), p2.setXRange(0, 6.28) ja p3.setXRange(0, 6.28).



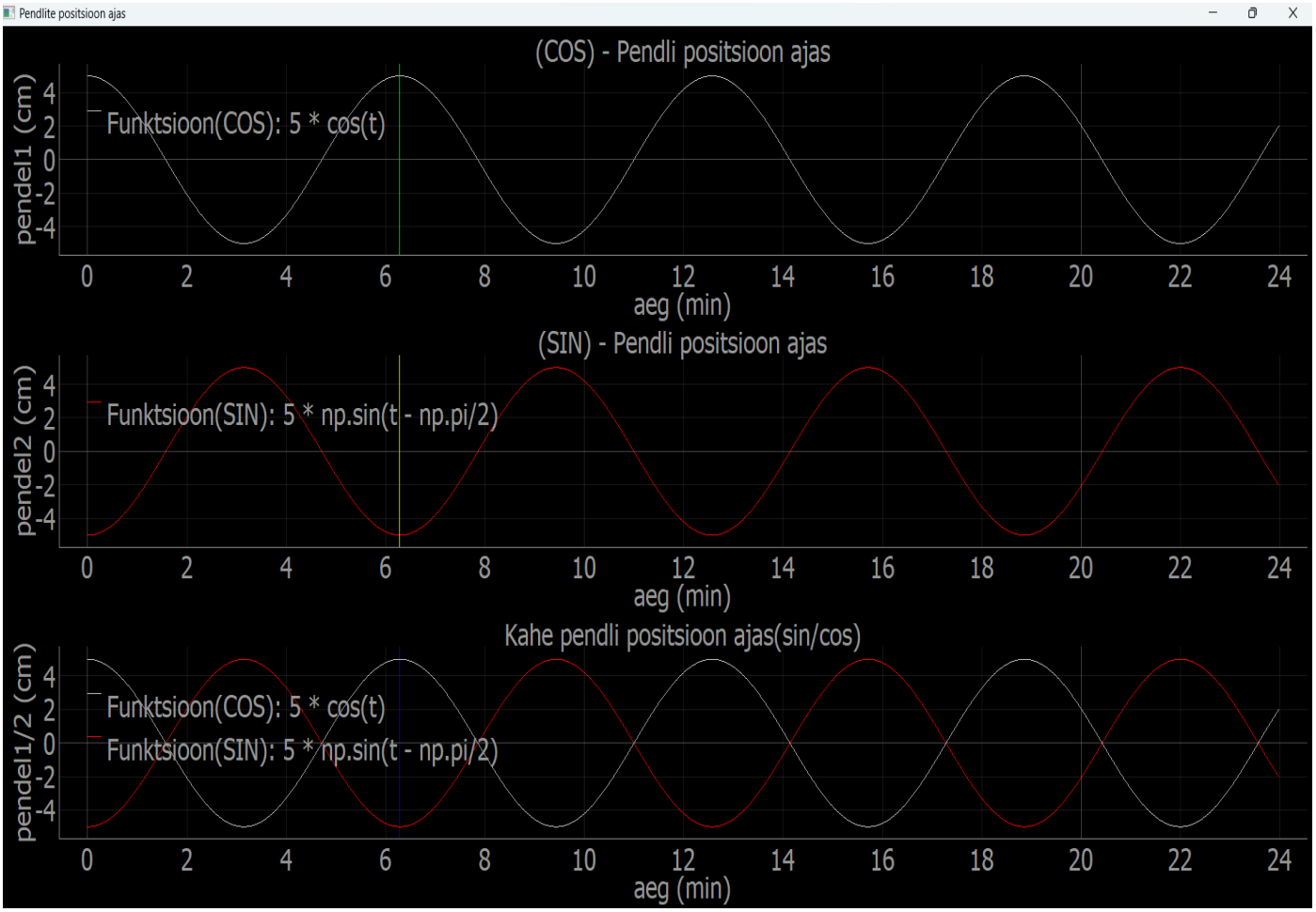
https://pyqtgraph.readthedocs.io/en/latest/api_reference/graphicsItems/plotitem.html#pyqtgraph.PlotItem.addLine

Teiseks valisin addLine(x=None, y=None, z=None, **kwds) funktsiooni. See funktsioon võimaldab minul

lisada graafikule horisontaalse või vertikaalse joone. Selleks tuleb esmalt määrata kas minul on soov x või y koordinaadi joone järele ning joone värv. See funktsioon on mulle kasulik, kui soovin tuua mingit kindlat väärtust välja graafikul. Eelmises alampunktis oli mul vaja arvutada graafiku periood. Tänu addLine funktsioonile saan valida eraldi värviga vertikaalse(x) joone, mis näitab mulle funktsiooni perioodi ehk korduva muutuse tsükli kestvust 0 punktist alates. Selles funktsioonis sain valida ka joone

värvi. Joone värv aitab mul paremini eristada, mis graafikuga on tegu ning paremini oma andmeid välja tuua graafikul.

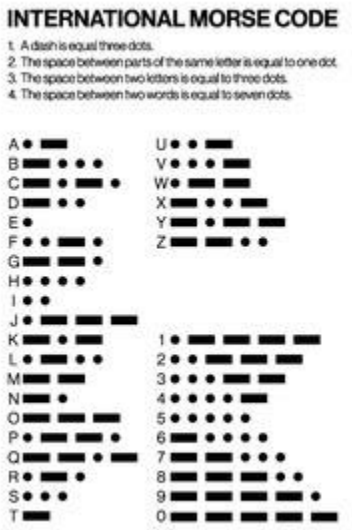
Lisasin enda varasemalt kirjutatud 1. Kodutöö_v1 koodi yl2.2(): koodiread: p1.addLine(x=6.28, pen="g"), p2.addLine(x=6.28, pen='y') ja p3.addLine(x=6.28, pen='b')



3. Ülesanne

https://et.wikipedia.org/wiki/Morse

Morse kood on rahvusvaheline side keel, mis koosneb punktidest, kriispudest ja pausidest.



Morse koodi visuaalselt kujutiselt näeme, et ladina tähestiku 26 tähele ning 10 põhi numbrile on määratud punktide ja kriipsude kombinatsioonid.

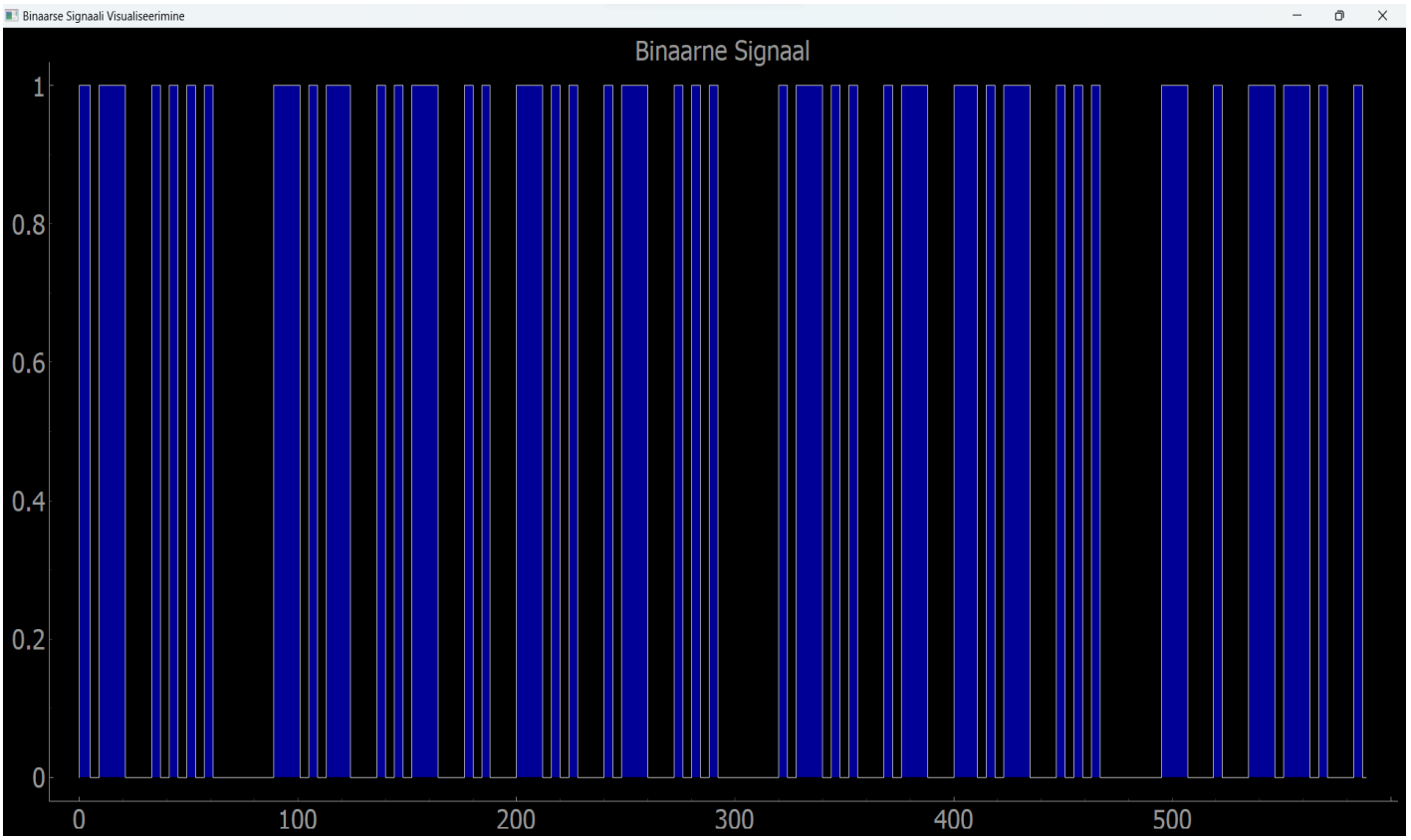
Punkt on lühike signaal ehk üks ajaperiood.

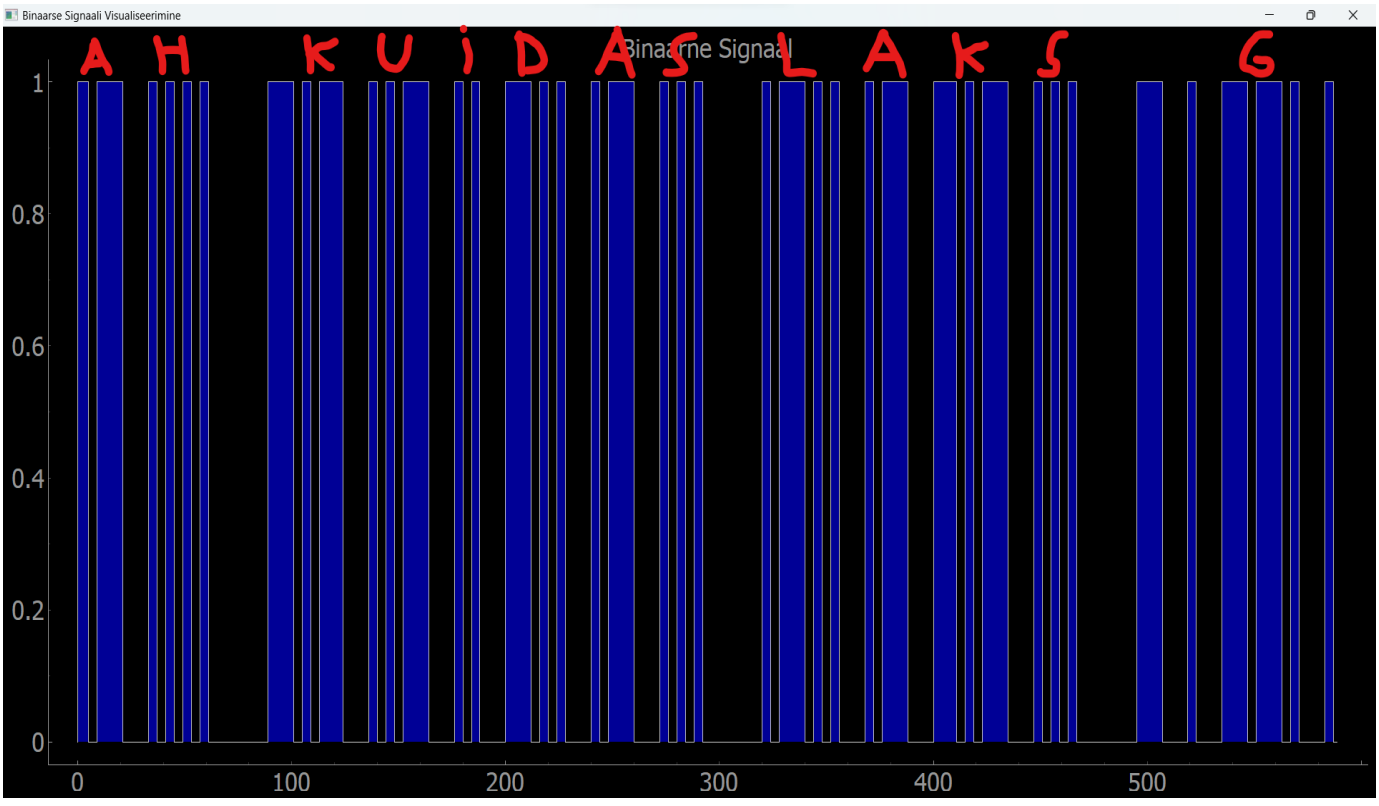
Kriipsu signaal on kolm korda pikem kui punkt ehk kolm ajaperioodi.

Tähtede vaheline vaikus kestab ühe ajaperioodi.

Sõnade vaheline vaikus kestab seitse ajaperioodi.

Selles ülesandes ma oletan, et signaali väärtus on kõrge arvuga 5 ja madal 0. Kõik muud 0-Ii lähedased arvud ntks 0.034 tähendab samuti 0-li. Kõrge signaali väärtus (5) tähendab signaali punkti või kriipsu. Madala signaali väärtus (0) esindab vaikust. Sõnade vaheline vaikus kestab seitse ajaperioodi ning on kõvasti pikem kui tähtede vaheline vaikus, mis kestab ühe ajaperioodi


Binaarse Signaali Visualiseerimine
Binaarne Signaal
1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500


Binaarse Signaali Visualiseerimine
AH KUIDAS LAKS G
Binaarne Signaal
1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500

Sain dekodeeritud sõnumiks „Kuidas Laks“.

Link minu repositooriumis olevale koodile, millega visualiseering loodi:

https://gitlab.ut.ee/loti.05.064/dsp-loti.05.064-ralf-suss-b31108-23-24k/-/blame/tudeng/Ralf-Suss/praktikumid/p01.py#L349

https://gitlab.ut.ee/loti.05.064/dsp-loti.05.064-ralf-suss-b31108-23-24k/-/blob/tudeng/Ralf-Suss/praktikumid/p01.py?ref_type=heads