Signaalide lähendamine sinusoididega

Sisukord

Aastal 2021 on vastutav juhendaja: Meelis Pihlap <pihlap@ut.ee> 2021-03-16|17:47:28 +0200

1. Ülevaade

Juhendist on olemas ka PDF versioon.

Tänases praktikumis vaatame signaale nii aja- kui ka sagedusruumis. Praktikumi lõpuks oskate liigutada signaali ajadomeeni ja sagedusruumi vahel ning kujutada oma tulemusi graafiliselt.

Juhendiga töötades pidage silmas

Praktikumijuhend sisaldab praktikumiülesannete kirjeldusi koos abistava infoga. Sealhulgas on toodud ka juhised oma süsteemi seadistamiseks. Töö õnnestumisele kaasa aitamiseks oleme teinud ka hulga märkmeid probleemsete kohtade osas. Selle info oleme jaganud tüübi järgi järgmise kolme kategooria vahel.

Siin toodud info võib aidata ülesande kiiremini või kavalamalt lahendada.

Siin toodud info võib aidata veaohtlikku kohta vältida.

Siin toodud info võib aidata vältida kahju riistvarale, tarkvarale või iseendale.

Töövahendid

Iga teegi järel on toodud klassi arvutites oleva paki versioon. See ei tähenda, et peate täpselt sama versiooni peale oma lahendusi kirjutama, küll aga tasuks jääda sama põhiversiooni piiresse ning kui on valik, siis pigem klassis kasutatava versiooni kanti.

1. Python (3.6.9)
2. NumPy (1.13.3)
3. PyQt (5.2)

Töö esitamine praktikumi lõppedes

Soovitame tungivalt hiljemalt iga alamülesande lõpetamise järgselt lisada uus versioon lahendusfailist oma harusse. GitLab keskkonnas on vaja luua *Merge request* ainult üks kord praktikumi jooksul.

1. Alustuseks veenduge, et olete oma kohaliku giti kausta **sees**
   a. Aktiivse kataloogi kontrollimiseks saate kasutada käsku `pwd` ning muutmiseks käsku `cd`
2. Kontrollige, et olete harul <**tudeng/eesnimi-perenimi**>
   a. Kasutage selleks käsku `git branch -l`
3. Sisestage käsk `git status`, et näha, kas on veel registreerimata muudatusi
   a. Kui mõni fail on muutunud, siis saate uue versiooni registreerimiseks kasutada käske `git add -p <muutunud-fail>` ja `git commit`
4. Kasutage käsku `git push`, et laadida kohalikud versioonid serverisse
   a. Lugege kindlasti käsu väljundit ning kahtluse korral kontrollige, kas failid jõudsid serverisse
5. Praktikumi lõppedes navigeerige veebilehele https://gitlab.ut.ee ning registreerige menüüde kaudu *Merge request* suunaga oma harust <**tudeng/eesnimi-perenimi**> harusse **master**
   a. Enne järgmist praktikumi vaatavad juhendajad teie pakutud versioonid üle ja liidavad need sobivuse korral master-nimelisse harusse

Enne alustamist

Repositooriumisse on lisatud uut sisu. Selle sisu kasutamiseks peate ta oma harusse liitma.

1. Alustuseks veenduge, et olete oma kohaliku giti kausta **sees**
   a. Aktiivse kataloogi kontrollimiseks saate kasutada käsku `pwd` ning muutmiseks käsku `cd`

2. Navigeerige master-nimelisele harule
    a. Kasutage selleks käsku `git checkout master`
3. Laadige alla ning liitke serveri versioon
    a. Kasutage selleks käsku `git pull`. Pange tähele ka käsu väljundina antud infot
4. Navigeerige tagasi oma harule
    a. Kasutage selleks käsku `git checkout tudeng/eesnimi-perenimi`. Muutke kindlasti käsus kasutatav haru nimi
5. Liitke kõik master-nimelisel harul olevad muudatused oma harule
    a. Kasutage selleks käsku `git merge master`

2. Sissejuhatus - Signaal kui sinusoidide summa

Keeruliste signaalide esitamine lihtsamate signaalide summana on signaalitöötluses (ja matemaatilises analüüsis üldisemalt) laialdaselt levinud võte.

Eelnevalt õppisime, et diskreetseid signaale on võimalik esitada nihutatud ja skaleeritud ühikimpulsside summana. Selles praktikumis kohtleme signaali mitme eraldiseisva sinusoidi summana. Teades, milliste omadustega (sagedus, amplituud, faas, vertikaalnihe) sinusoididest signaal koosneb, on võimalik signaali taasluua. Selleks on aga tarvis need omadused kõigepealt välja uurida.

Praktikumi vältel omandame intuitiivse arusaama sellest, kuidas lähendada signaale sinusoidide summana ja implementeerime jupphaaval programmi, mis suudab eraldada tundmatust signaalist sinusoidid, millest see koosneb.

Kõigepealt tuletage meelde, millised on sinusoidi omadused diskreetses ja pidevas ajas vaadates järgnevat videot:

▶ https://www.youtube.com/watch?v=fCAZ7jcO-vc *(YouTube video)*

3. Kohustuslikud ülesanded

1. Ülesanne: Sagedus ja amplituud

Võtame eesmärgiks välja selgitada tundmatu signaali sageduse ja amplituudi. Selleks genereerime ise erinevate amplituudide ja sagedustega sinusoide ja võrdleme neid visuaalselt algsignaaliga.

Ava koodifail praktikum4.py, sisesta muutuja `STUDYBOOK_NR` väärtuseks oma matriklinumber ja `TASK_NR` muutuja väärtuseks 1. Käivita programm ja tutvu kasutajaliidesega.

1.1 Signaaligeneraatori implementeerimine

Tõenäoliselt märkasite, et liugurite liigutamine ei tee hetkel midagi ja ka üksiku komponendi graafik ei meenuta eriti sinusoidi. Implementeeri failis `praktikum4.py` funktsiooni `f` sisu nii, et funktsioon rakendaks sisendandmetele koosinusfunktsiooni, mis teeb antud ajavahemikus `k` perioodi amplituudiga `A` ning tagastaks tulemuse NumPy massiivina. Tulemuste graafikule kandmine on aluskoodis juba lahendatud.

Selle etapi tulemusena peaks programmi väljund nägema välja umbes selline:

▶ ./pildid/yl1.mp4 *(video)*

1.2 Signaaligeneraatori väljundi tõlgendamine

Vasta järgnevatele küsimustele.

1. Mitme sämpli põhjal me signaalikomponente arvutame?
2. Võrdle funktsiooni kuju kohtadel k=2 ja k=198. Mida märkad? Mis võiks olla põhjuseks?
    - Vastamiseks tutvu järgmise animatsiooniga, mis on võetud siit veebimaterjalist.

▶ ./pildid/aliasing.mp4 *(video)*

1.3 Katse-eksituse meetod

Nüüd on võimalik välja selgitada algsignaali sagedus ja amplituud, katsetades sageduse ja amplituudi kombinatsioone, kuni leiame sellise, mis kattub täiuslikult algsignaaliga. Nüüd, kus esimene signaal on uuritud - muuda muutuja `RANDOM_SEED` väärtust alguses kaheks ja siis kolmeks, ning leia tekkivate algsignaalide sagedused ja amplituudid. Seejärel ole valmis tegema sama juhendaja valitud sisendiga.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

2. Ülesanne: Nihked

Määra muutuja `TASK_NR` väärtuseks 2 ja `RANDOM_SEED` väärtuseks 1.

2.1 Parem signaaligeneraator

Täienda funktsiooni `f`, rakendades vertikaalnihet ja faasinihet. Selle etapi tulemusena peaks programmi väljund nägema välja umbes selline:

▶ ./pildid/yl2.mp4 *(video)*

2.2 Katse-eksituse meetod

Signaaligeneraatorisse lisatud uute parameetrite abil oleme võimelised ka tundmatus signaalis neid parameetreid hindama. Muuda muutuja `RANDOM_SEED` väärtust kolmel korral ja leia tekkivate algsignaalide sagedused, amplituudid, vertikaalnihked ja faasinihked. Seejärel ole valmis tegema sama juhendaja valitud sisendiga.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

3. Ülesanne: Sinusoidide summa

Määra muutuja `TASK_NR` väärtuseks 3.

Nagu mainitud praktikumi alguses, on mistahes funktsiooni võimalik esitada sinusoidide summana. Rakendame seda teadmist, et uurida veel keerulisemaid signaale.

3.1 Veel parem signaaligeneraator

Keerulisemaid signaale saame tekitada liites omavahel mitu sinusoidi. Implementeeri funktsiooni signal_sum sisu. Selle etapi tulemusena peaks programmi väljund nägema välja umbes selline:

▶ ./pildid/yl3.mp4 *(video)*

3.2 Katse-eksituse meetod

Muuda muutuja `RANDOM_SEED` väärtust kolmel korral ja leia tekkivate algsignaalide sagedused, amplituudid, vertikaalnihked ja faasinihked. Seejärel ole valmis tegema sama juhendaja valitud sisendiga ja vastama juhendaja küsimustele.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

4. Ülesanne: Piirjuhud

Määra muutuja `TASK_NR` väärtuseks 4, käivita programm ja leia algsignaali sagedus, amplituud, vertikaalnihe ja faasinihe.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

5. Ülesanne: Signaalide kattuvuse hindamine

Määra muutuja `TASK_NR` väärtuseks 5.

Silma järgi signaali komponentide sobitamine on tehtav, kui signaalikomponente on vaid kaks. Kui aga komponentide arv tõuseb kümnetesse või sadadesse, ei ole enam nende silma järgi leidmine mõeldav. Selliste signaalide puhul on mõistlik signaalide kattuvus välja arvutada.

5.1 Hinnang skalaarkorrutisena

Üks moodus kattuvuse hindamiseks on skalaarkorrutis (ingl *dot product*). Loe läbi esimesed kaks peatükki antud veebimaterjalist ja vasta järgnevatele küsimustele. Järgmise peatüki leiad vajutades lehe allosas **[Next]** nuppu

1. Mida tähendab, kui kahe signaali skalaarkorrutise väärtus on negatiivne?
2. Mida tähendab, kui kahe signaali skalaarkorrutise väärtus on null?
3. Kuidas mõjutab faasinihe skalaarkorrutise tulemust?
4. Kuidas mõjutab amplituudi või vertikaalnihke muutmine skalaarkorrutise tulemust?

5.2 Kattuvuse hindamine programmaatiliselt

Implementeeri kahe signaali skalaarkorrutis funktsioonis compare_signals. Selle tulemusena saame omale sobituse hetkeseisule juurde arvulise hinnangu.

Vältimaks olukorda, kus leitud kattuvuse suurus on sõltuv andmepunktide hulgast, on mõistlik tulemus andmepunktide arvuga läbi jagada.

Selle etapi tulemusena peaks programmi väljund nägema välja umbes selline:

▶ ./pildid/yl5.mp4 *(video)*

Muuda muutuja `RANDOM_SEED` väärtust kolmel korral ja leia tekkivate algsignaalide komponentide sagedused, amplituudid, vertikaalnihked ja faasinihked käsitsi liugureid liigutades. Selle tegevuse ajal jälgi, kuidas muutub skalaarkorrutise väärtus, kui õigetele signaalidele lähemale jõudma hakkate. Nüüd näita tulemust juhendajale ning tee sama tema valitud sisendiga.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

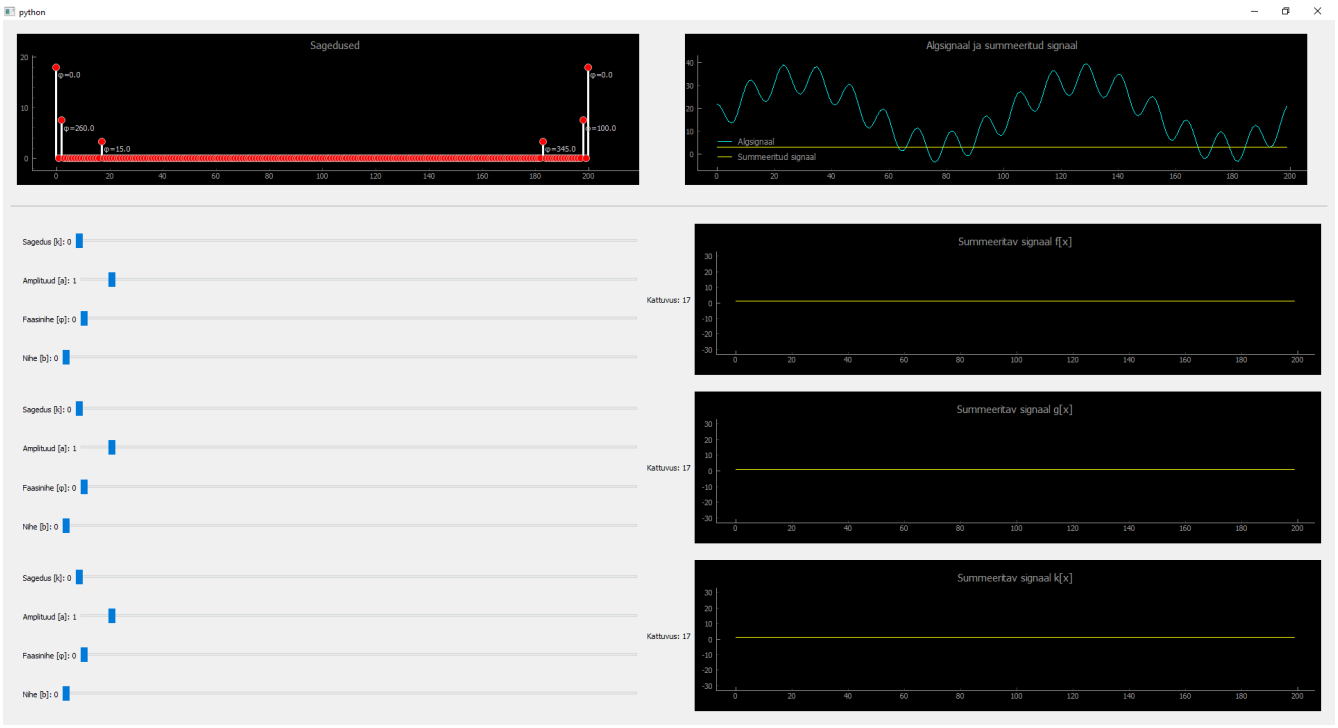
6. Ülesanne: Käsi on väsinud, kas seda kuidagi automatiseerida ei saa?

Määra muutuja `TASK_NR` väärtuseks 6.

Seni oleme implementeerinud signaaligeneraatori ja mooduse hindamiseks, mil määral on genereeritud signaal esindatud algses signaalis. Viimase sammuna implementeeri funktsioon `find_frequencies` ja leia aluskomponendid programmaatiliselt. Selle jaoks:

1. Käi iteratiivselt läbi kõik sageduste [0..N+1] ja faasinihete kombinatsioonid. Faasinihkel olgu sammu pikkuseks `(2pi)/360`.
2. Genereeri vastavate sageduste ja faasinihetega koosinussignaalid (amplituudiks määra kõigile hetkel 1 ja vertikaalnihkeks 0)
3. Võrdle algsignaali ja genereeritud signaali kasutades eelmises ülesandes implementeeritud funktsiooni
4. Leia iga sageduse kohta kõige kõrgema kattuvuse saavutav faasinihe
5. Tagasta NumPy massiiv, mille indeksid tähistavad vaadeldud sagedusi ja elementideks on kahe-elemendilised NumPy massiivid, kus esimene element on parim saavutatud kattuvus ja teine selle kattuvuse saavutanud faasinihe.

Selle etapi tulemusena peaks programmi väljund nägema välja umbes selline:



Viimaks näidake juhendajale liugurite abil, et algsignaal koosneb tuvastatud sagedustega komponentidest ja demonstreerige arusaamist sagedusgraafiku väljundist.

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

4. Kodutöö ülesanded

Selle kodutöö eesmärgiks on ehitada tarkvaraline süntesaator, mis suudab tekitada erineva tämbriga meloodiaid. Teisisõnu proovime luua oma isikliku kõlaga lihtsa pilli, millel saab mängida teie poolt valitud loost väikse viisijupi.

Teie loodud lahendus peab vastama järgnevatele nõutele:

1. Iga koodirida kodutöö failis peab olema põhjalikult kommenteeritud.
    a. `import` käskude seletamiseks ei pea olema detailsed kommentaarid.
2. Kõik kasutatavad muutuja- ning funktsiooninimed peavad olema läbi mõeldud.
3. Programmi käivitamisel peab programm looma koodiga samasse kausta kõik kolm .wav faili ilma lisasisendit vajamata.
    a. Kui juhendajad peavad teie koodi parandama või selles juppe sisse ja välja kommenteerima, olete selle punkti vastu eksinud.
4. Loodud `.py` faili nimi peab vastama eelnevates kodutöödes kehtestatud vormile.
    a. p04_kodutoo_perenimi_v1.py
5. Juhendajate kõrvade säästmiseks mitte kasutada põhihelidena kõrgemaid sagedusi kui 2000 Hz. Loodud kood tuleb laadida oma repositooriumi .py failina ning esitada sama sisu .txt failina Moodle keskkonda lingi alla "4. kodutöö esitamine".

1. ülesanne

Kirjutage programm, mis genereerib järgnevate parameetritega kuulatava sinusoidse helisignaali ja salvestab selle faili `yl1.wav`.

- Diskreetimissagedus 44100 Hz
- Bitisügavus 16-bit PCM
- Heli kestvus 2 sekundit
- Helisagedus 440 Hz

Veenduge Audacity abil, et loodud heli sagedus sai õige.

2. ülesanne

Kindlasti olete täheldanud, et reaalsete pillide kõla on keerulisema loomuga kui lihtsalt üks sageduskomponent.

- Täiendage esimeses ülesandes loodud programmi nii, et väljundsignaali tämber üritaks imiteerida mõnda instrumenti.
  - Inspiratsiooni selle jaoks leiab näiteks siit.
- Selgitage koodifailis, mis instrumenti te jäljendate.
- Salvestage tulemus faili `yl2.wav`.

3. ülesanne

Mitu helikõrgust esitatuna teineteise järel moodustavad viisi ehk meloodia. Täiendage programmi nii, et väljundsignaaliks on meloodia, mis on mängitud eelmises ülesandes implementeeritud instrumendi tämbriga.

- Meloodia peab sisaldama vähemalt kolme erinevat ja kokku vähemalt kümmet põhiheli(nooti).
- Kui ei soovi täna oma heliloojakarjääriga alustada, võib meloodiaid laenata näiteks siit
- Ülaltoodud lingil leiduvate tähtede tähenduste tõlgendamiseks tutvu vajaduse korral selle dokumendiga.
- Et kõrvuti paiknevad samade helikõrgustega noodid oleksid teineteisest eristatavad, tekita iga noodi lõppu hetk vaikust.
- Salvestage tulemus faili `yl3.wav`.