Mérési útmutató az
Intelligens beágyazott rendszerek laboratórium (VIMIMA21)

Elosztott adatgyűjtő rendszerek

című méréséhez

Készítette:
Dr. Orosz György
BME Méréstechnika és Információs Rendszerek Tanszék

2023. július

Tartalomjegyzék

Ábrák jegyzéke

5. mérés

Elosztott rendszerek és szenzorhálózatok

5.1. A mérés célja

Manapság egyre több területen találkozhatunk különféle vezeték nélküli rendszerrel. Ezek a rendszerek igen nagy flexibilitást biztosítanak a vezeték nélküli adattovábbítás miatt, és általában különféle információgyűjtési és feldolgozási feladatot látnak el. Mivel a vezeték nélküli hálózatok egységei (a szenzor node-ok) önálló működésűek, így a rendszer elosztott működésűnek tekinthető. Az elosztott rendszerek egyik nagy előnye, hogy az egyes egységek általában önálló vezérlőegységgel rendelkeznek, melyek akár lokális döntéseket képesek hozni, így teszik hatékonyabbá a rendszer működését.

A vezeték nélküli kommunikáció és az elosztott jelérzékelés és feldolgozás azonban több, alapvető problémát is felvet. A mérés célja, hogy megismertesse a hallgatókat az elosztott, vezeték nélküli jelfeldolgozó rendszerek jellegzetes jelfeldolgozási problémáival. A mérés során egy egyszerű, vezeték nélküli szenzorokra épülő rendszeren keresztül nyerünk betekintést különböző jelfeldolgozási problémákba. A vezeték nélküli hálózatot mitmótok alkotják, melyek egy, a 2.4 GHz-es sávban működő, CC2420 típusú kommunikációs IC-t használnak a rádiós adatátvitelhez.

A mérés két nagy témakör köré épül:

1. Szinkronizáció
2. Rezonátor alapú adattömörítés (compressive sensing)

A felhasznált szenzorhálózat egy PC-hez kapcsolódik, mely a szenzorok által gyűjtött adatok archíválására és feldolgozására szolgál.

A szinkronizáció problémája végigvonul a mérés során. Az első mérési feladatban egy egyszerű adatgyűjtő rendszer esetén ismerkedhetünk meg egy úgynevezett időbélyeg-transzformációt alkalmazó szinkronizációs technikával.

Második fő feladat egy rezonátor alapú megfigyelőt alkalmazó adattömörítést használó rendszer megismerése. Ebben az esetben egy vezeték nélküli szenzor előállítja a megfigyelt jel Fourier-együtthatóit, és csupán ezeket továbbítja a rádiós csatornán. Mivel ezek az együtthatók általában lassabban változnak mint maga a jel, így ritkábban is továbbíthatóak, ezzel valósítva meg az adattömörítést. A szinkronizáláson kívül ebben a rendszerben meg kell valósítani a jel visszaállítását a mért paraméterekből.

A mérés során a hallgatók gyakorlatot szerezhetnek jelfeldolgozási feladatok megvalósításában, és az elosztottan végzett megfigyelések értelmezésében, valamint a felmerülő hibák korrigálásában.

5.2. Szinkronizáció

5.2.1. Időbélyeg transzformáció megvalósítása

A szinkronizáció alapvető szerepet játszik az elosztott, vezeték nélküli rendszerekben. Ezekben az esetetekben az egyes alegységek igen laza kapcsolatban állnak egymással a vezeték nélküli csatornán keresztül, emiatt autonóm egységeknek tekinthetőek, melyek nem állnak *közvetlen* központi felügyelet alatt.

Tipikus és alapvető feladat a vezeték nélküli rendszerek esetén bizonyos analóg jelek (fényerő, hőmérséklet, hang ...) megfigyelése. A mérések kiértékelése azonban megfelelő időinformációt is feltételez a megfigyelések mellett. Például egy objektum haladási sebességének mérésekor nem elegendő csupán a helyzetének ismerete, hanem mindenképpen ismerni kell, hogy az objektum adott helyen milyen időpontokban tartózkodik. Mivel elosztott rendszerek esetén az egyes egységek általában önálló órával rendelkeznek, így az időinformációk szolgáltatása nem triviális feladat, mely főleg gyorsan változó jelek esetén jelent nagy kihívást.

Az időmérés alapjául általában valamilyen kvarcoszcillátor szolgál, mely igen jó kompromisszum az ár, pontosság és stabilitás szempontjából. (Ilyen kvarcoszcillátorok találhatóak például sok karórában is.) A nagyságrendek érzékeltetése szempontjából fontos tudni, hogy egy általános célú kvarc hibája maximum néhány tíz 10 ppm körül várható (ppm=parts per million=10^{-6}). Ez azt jelenti, hogy pl. 10 ppm hiba esetén egy 10 MHz-es oszcillátor a névleges értéktől maximum $\pm$100 Hz-el tér el. Még ez a relatív kis hiba is jelentős eltérést okozhat az időmérésben hosszú idő alatt. Például, ha feltételezzük, hogy két szenzor órája a nap elején megegyezik, viszont a bennük található kvarcok frekvenciáinak eltérése 10 ppm, akkor nap végére az óráik által mutatott idő különbsége eléri a $24 \cdot 60 \cdot 60 \cdot 10 \cdot 10^{-6} = 0.864$ sec-t, tehát majdnem egy másodperces eltérés tapasztalható. Precíz mérések esetén ekkora hiba elfogadhatatlan.

A szenzorok szinkronizálására egy kézenfekvő módszer az órák folytonos hangolása (például a déli harangszó hallatára beállítjuk a karóránkat pontosan délre). Az órák hangolása azonban nem mindig lehetséges. Ebben az esetben használható például az időbélyeg-transzformációt megvalósító szinkronizációs algoritmus. Időbélyegnek nevezzük azt az időadatot, melyet valamilyen eseményhez rendelünk, pl. mikor történt egy megfigyelés.

Az algoritmus megismerése előtt néhány definíció tisztázása szükséges. A továbbiakban t jelöli a referencia időt, mely az abszolút pontos időt jelenti (pl. atomóra által szolgáltatott idő). $h_i(t)$ jelöli az i-edik node órája által mutatott időt. $O_{i,j}(t) = h_i(t) - h_j(t)$ az i-edik és a j-edik node órája közötti ofszet, tehát időkülönbség. $\rho_{i,j}$ az i-edik és a j-edik node órája közötti drift (elcsúszás), mely megadja, hogy a két óra közötti különbség mennyit változik egységnyi idő alatt: $\rho_{i,j} = \frac{dO_{i,j}}{dt} \approx (O_{i,j}(t_2) - O_{i,j}(t_1))/(t_2 - t_1)$. Ez utóbbi becslés akkor igaz, ha a két óra egyenletesen csúszik el egymáshoz képest, tehát a kvarcoszcillátorok frekvenciahibája állandó. Ez általában (legalább rövid távon) igaz, ekkor a drift megegyezik a kvarcok hibájával. Definiálhatjuk még az órák sebességét (gyorsaságát): $f_i = \frac{dh_i}{dt}$, mely azt mutatja meg, hogy milyen gyorsan telik az idő az i-edik node-on a referenciaidőhöz képest. Ideális esetben $f_i = 1$, tehát az óra egységnyi sebességgel jár. Az egymáshoz képest mért sebesség is definiálható hasonló módon: $f_{i,j} = \frac{dh_i}{dh_j}$.

Példa 1: Két óra (node_1 és node_2) által mutatott időpontokat feljegyeztük a következő időpontokban:

[2000. jan. 10. 12:00:00]-kor

- a node_1 órája: [2000. jan. 10. 12:00:10]-t mutat
- a node_2 órája: [2000. jan. 10. 12:00:09]-t mutat.

[2000. jan. 15. 12:00:00]-kor

– a node$_1$ órája: [2000. jan. 15. 12:00:11]-t mutat
– a node$_2$ órája: [2000. jan. 15. 12:00:08]-t mutat.

Ebben a példában a referencia idő az első esetben: t_1 =[2000. jan. 10. 12:00:00], tehát:

t_1 =[2000. jan. 10. 12:00:00] időpontban:

– az első node lokális ideje: $h_1(t_1)$ =[2000. jan. 10. 12:00:10]
– a második node lokális ideje: $h_2(t_1)$ =[2000. jan. 10. 12:00:9]
– a két óra közötti ofszet tehát: $O_{1,2}(t_1) = h_1(t_1) - h_2(t_1) = 1$ sec

A referencia idő a második esetben: t_2 =[2000. jan. 15. 12:00:00], tehát:

t_2 =[2000. jan. 15. 12:00:00] időpontban:

– az első node lokális ideje: $h_1(t_2)$ =[2000. jan. 15. 12:00:11]
– a második node lokális ideje: $h_2(t_2)$ =[2000. jan. 15. 12:00:8]
– a két óra közötti ofszet tehát: $O_{1,2}(t_2) = h_1(t_2) - h_2(t_2) = 3$ sec.

A két időpont közötti időkülönbség: $\Delta t = t_2 - t_1 = 5$ nap $= 432000$ sec. A két node órája közötti különbség tehát $(3-1)$ sec-t változott Δt idő alatt, tehát a két óra közötti drift (elcsúszás sebessége): $\rho_{i,j} = (3-1)$ sec$/\Delta t = 4.63 \cdot 10^{-6} = 4.63$ ppm.

Itt jegyezzük meg, hogy az óráknak nem fontos a hagyományos értelemben vett időt és dátumot mutatni, lehet ez az időinformáció például a bekapcsolásuktól számított idő, mint ahogyan az a mérésben használt rendszer esetében látni fogjuk.

Folytatva a példát, tegyük fel, hogy a két óra egy híd két végén van elhelyezve, és mérik, hogy egy adott objektum mennyi idő alatt halad át a hídon. Az 1. óra azt jelentette, hogy egy autó $h_1(t_3)$=[2000. jan. 25. 12:00:13]-kor hajtott rá a hídra a 2. óra pedig azt jelentette, hogy az adott autó $h_2(t_4)$=[2000. jan. 25. 12:00:11]-kor hajtott le. Mennyi idő alatt hajtott át az autó?

A megoldáshoz közös időskálára kell hoznunk a két órát, tehát el kell végeznünk egy időbélyeg-transzformációt. Az előző számításokból tudjuk, hogy [2000. jan. 15. 12:00:00]-kor az órák közötti ofszet $O_{1,2} = 3$ sec valamint láthatjuk, hogy az ofszet 5 nap alatt 2 sec-t változik, tehát [2000. jan. 25. 12:00:13]-kor az ofszet kb. 3 sec $+ 2 * 2$ sec $= 7$ sec. Ezek szerint, ha az első óra időskáláját vesszük alapul, akkor megállapíthatjuk, hogy az autó az első óra szerint $h_1(t_4) = h_2(t_4) + 7$ sec=[2000. jan. 25. 12:00:18]-kor hajtott le a hídról, ami azt jelenti, hogy az autó $h_1(t_4) - h_1(t_3) = 5$ sec alatt haladt át a hídon. (Megjegyezzük, hogy itt tettünk néhány ésszerű egyszerűsítést: pl. nem vesszük figyelembe, azt, hogy a két óra valamennyit elcsúszik egymáshoz képest annyi idő alatt is, míg az autó áthalad a hídon, de ez ebben az esetben nagyságrendi hibát nem okoz.) Láthatjuk, hogy mivel az órák nem ideálisak, így egymástól függetlenül járnak, tehát nem lehetséges az általuk mutatott időadatok (időbélyegek) közvetlen felhasználása, hanem valamilyen szintű átalakítás, tehát időbélyeg-transzformáció szükséges. A következő részben ennek a transzformációnak a matematikai alapjai kerülnek bemutatásra, melyek felhasználásával egyszerű módon végrehajtható a transzformáció.

Általánosságban elmondható, hogy két egység esetén az egyes egységek lokális idejei felírhatóak egymás függvényeként:

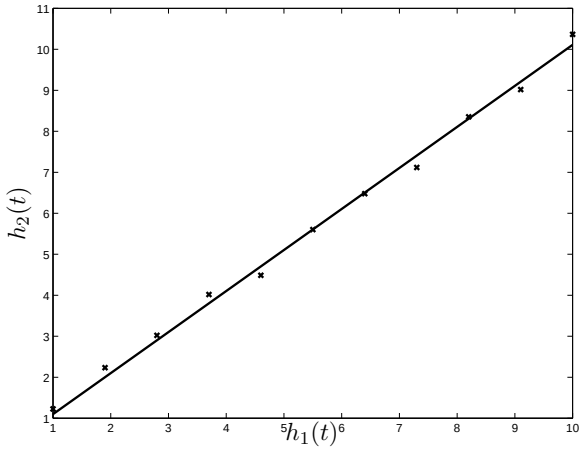
$$h(t)_i = g(h_j(t)). \tag{1}$$

Egy gyakorlatban is jól használható összefüggést kapunk, ha feltételezzük, hogy a lokális idők egymásnak lineáris függvényei, ekkor:

$$h_i(t) = a h_j(t) + b, \tag{2}$$

A lineáris függvény azt jelenti, hogy azt vesszük figyelembe, hogy az órák kvarcoszcillátorainak frekvenciahibája állandó. Ez általában, legalábbis rövid távon igaz. Amennyiben feltételezzük, hogy (2) igaz, és ismerjük az a és b paramétereket, akkor az i-edik és j-edik node órája között az átváltást egyértelműen elvégezhetjük. Ezt a műveletet nevezzük időbélyeg-transzformációnak. Ezzel elérhetjük, hogy konziszens időeredményeket kapjunk és két, egymástól különálló szenzor által érzékelt eseményeket értelmezni tudjuk. Természetesen több egység esetén is alkalmazható a módszer, ebben az esetben ki kell választani, hogy mi legyen a közös időskála.

(2) egyenletben található paraméterek meghatározhatóak úgynevezett szinkronizációs pontok segítségével, melyek ugyanazon időponthoz tartozó időbélyeg párok. A példánkban ezek az időbélyeg párok: $\{h_1(t_1), h_2(t_1)\}$ valamint $\{h_1(t_2), h_2(t_2)\}$. Két szinkronizációs pont alapján már meghatározható (2) egyenlet két paramétere: a és b. A paraméterek meghatározásához általában több szinkronizációs pontot használunk, így a mérési hibák hatása csökken. Elsőfokú egyenlet esetén, mint például (2), a paraméterek meghatározásához használható a lineáris regresszió. Erre látható példa az 1. ábrán. A MATLAB hatékony segítséget nyújt a különböző fokú görbeillesztéshez a `polyfit` parancs segítségével.



1. ábra. Lokális idők függvényei és szinkronizációs pontok. Folytonos vonal: $h_2(t) = ah_1(t) + b$, $\times$: $\{h_1(t_k), h_2(t_k)\}$ szinkronizációs pontok.

Általánosságban tehát elmondható, hogy az időbélyeg-transzformáció elvégzése három lépésben hajtható végre:

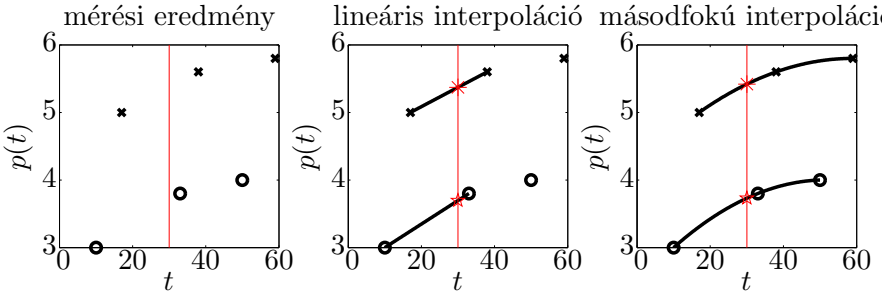
1. Szinkronizációs pontok gyűjtése.
2. A szinkronizációs pontok alapján meghatározzuk az időbélyegek átváltásához szükséges paramétereket: a és b a (2)-es egyenletben.
3. Az időbélyegek konverziója az előző pontban meghatározott paraméterek és (2) alapján.

5.2.2. Szinkronizáció interpolációval

Az önálló egységek szinkronizálatlansága mellett az elosztott jelérzékelés másik problémája, hogy az egyes eszközökön a megfigyelés, tehát a jelek mintavételezése egymástól többé-kevésbé függetlenül zajlik. Sok esetben azonban nem csak korrekt időbélyegekkel ellátott adatokra, hanem időben összetartozó adatokra van szükség.

Példa 2: Két szenzor (node_1 és node_2) két tartály nyomását ($p_1(t)$ és $p_2(t)$) méri egymástól függetlenül. Egy adott percben a következő mérési eredmények születtek az első

tartály esetén: {17 sec, 5 bar}, {38 sec, 5.6 bar}, {59 sec, 5.8 bar}. A másik tartány esetén: {10 sec, 3 bar}, {33 sec, 3.8 bar}, {50 sec, 4.0 bar}. A grafikonok a 2. ábrán láthatóak. Mekkora a nyomáskülönbség a tartályban adott perc $t_p = 30$ sec-kor. (Tegyük fel, hogy az időskála már helyes.)



2. ábra. Mért nyomásértékek, valamint becslés első és másodrendű interpolációval. ×: $p_1(t)$, ○: $p_2(t)$

Látható, hogy $t_p = 30$ sec-kor nincsen mérési eredmény, így a nyomásértékeket valahogyan becsülni kell. Erre egy kézenfekvő módszer valamilyen polinom illesztése a mérési pontokra. Erre látható példa a 2. ábrán. Legegyszerűbb eljárás a lineáris interpoláció. Ebben az esetben a mérési eredmények között a mért jelet egy egyenessel becsüljük. Ennek matematikai felírása a következő. Legyen adott egy mennyiség értéke T_1 és T_2 időpontokban: $x(T_1)$ és $x(T_2)$. Legyen a T_1 és T_2 közötti különbség: $T_p = (T_2 - T_1)$. Az x mennyiséget a következő módon becsülhetjük $t_p \in [T_1, T_2]$ időpontban:

$$x(t_p) = x(T_1)\frac{T_2 - t_p}{T_p} + x(T_2)\frac{t_p - T_1}{T_p}. \tag{3}$$

Tehát az egyes mérési eredményeket annak arányban súlyozzuk, hogy a kérdéses t_p időpont milyen messze van a mintavételi időpontoktól.

Példánkban a módszer a következő eredményt adja: $p_1(30) = 5 \text{ bar} * \frac{38-30}{38-17} + 5.6 \text{ bar} * \frac{30-17}{38-17} = 5.37$ bar valamin $p_2(30) = 3 \text{ bar} * \frac{33-30}{33-10} + 3.8 \text{ bar} * \frac{30-10}{33-10} = 3.7$ bar.

A becslésre természetesen magasabb fokszámú polinomok is használhatóak. Ebben az esetben a kérdéses időpont környezetében található pontokra polinomot illesztünk, amit kiértékelünk a kérdéses helyen. Magasabb fokú polinom illesztéséhez jól használhatóak a MATLAB `polyfit` és `polyval` nevű függvényei.

A `polyfit` függvénynek meg kell adni azokat az `x` és `y` pontokat, amelyekre egy `n`-ed fokú polinomot szeretnénk illeszteni, és a függvény visszaadja az illesztett polinom együtthatóit egy `coef` vektorban: `coef = polyfit(x,y,n)`.

Az előző példában másodfokú interpolációt használva a következő MATLAB kódot használhatjuk másodfokú polinom illesztésére:

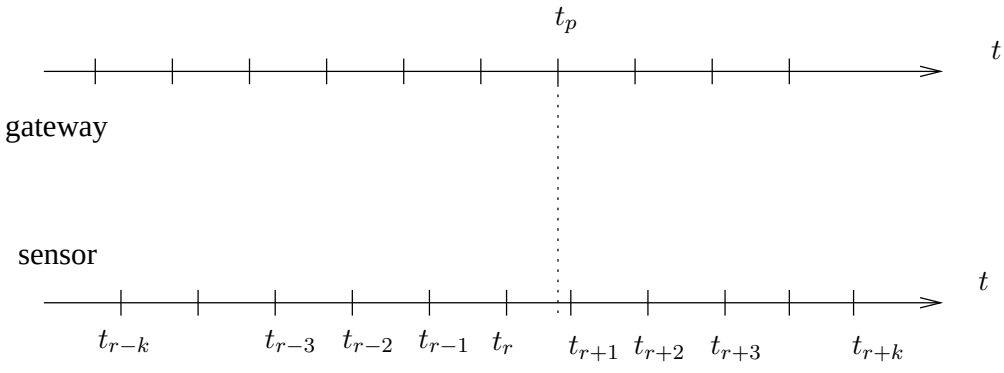
```
illesztPoly_1 = polyfit([17 38 59],[5 5.6 5.8],2);
           %polinom illesztése ismert pontokra
nyomas_1 = polyval(illesztPoly_1,30) %az illesztett polinom
          %kiértékelése a kívánt pontban
nyomas_1 =
     5.4186

illesztPoly_2 = polyfit([10 33 50],[3 3.8 4.0],2);
```

```
nyomas_2 = polyval(illesztPoly_2,30)
nyomas_2 =
      3.7302
```

Megjegyezzük, hogy egy n-ed rendű polinom illesztéséhez legalább $n+1$ pontot kell megadni a `polyfit` függvénynek. A mérés során előfordulhat, hogy az időértékek különbsége értékükhöz képest relatíve kicsi. Például ha az időértékek a második esetben:
[10000000010 10000000033 10000000050] lennének a kérdéses időpont pedig 10000000030, akkor numerikus okok miatt a számítás igen pontatlan; erre a MATLAB figyelmezteti is a felhasználót. Ebben az esetben két megoldás javasolható: el kell tolni az értékeket például az első elemet kivonva mindegyikből, hogy a különbségük relatíve nagy legyen, valamint átskálázhatjuk (megszorozzuk egy konstanssal), így egy elfogadható nagyságrendbe kerülnek az értékek. Ezzel nem változik meg az illesztés utáni érték, hiszen az egész függvényt csupán eltoljuk az x tengely mentén illetve arányosan nyújtjuk. Ekkor a következő kódot kapjuk:

```
t=[10000000010 10000000033 10000000050] %mérés időpontjai
tp=10000000030 %kérdéses időpont, ahol becsüljük a függvényt
x=[3 3.8 4.0]
scaleF=1000; %skálatényező: ezzel szorozzuk az x tengelyt
          %illesztésnél és számításnál is
illesztPoly_2 = polyfit( (t - t(1))*scaleF, x, 2 );
nyomas_2 = polyval( illesztPoly_2, (tp-t(1))*scaleF )
nyomas_2 =
      3.7302
```

A fenti kódban az egyszerűség és általánosság kedvéért az időpontokat a `t` és `tp` változóba helyeztük, az eltolást pedig `t(1)`-gyel végeztük, ami a `t` vektor első eleme. Skálatényezőként `scaleF`-t alkalmaztuk. Látható, hogy a végeredményt sem az eltolás sem a skálázás nem befolyásolja. Amennyiben a MATLAB ezen megoldások (skálázás, eltolás) gyengén kondícionáltnak tartja problémát, a figyelmezetető üzeneteket a
`warning('OFF', 'MATLAB:polyfit:RepeatedPointsOrRescale');` paranccsal kapcsolhatjuk ki. Figyelem! Ez nem szünteti meg a problémát, csupán a hibaüzeneteket kapcsolja ki, melyeknek gyakori megjelenítése nagyban lassítja a program futását. Csak akkor használjuk, ha más módszerrel nem sikerült megoldani a problémát!



3. ábra. Interpolációs algoritmus időzítési diagramja

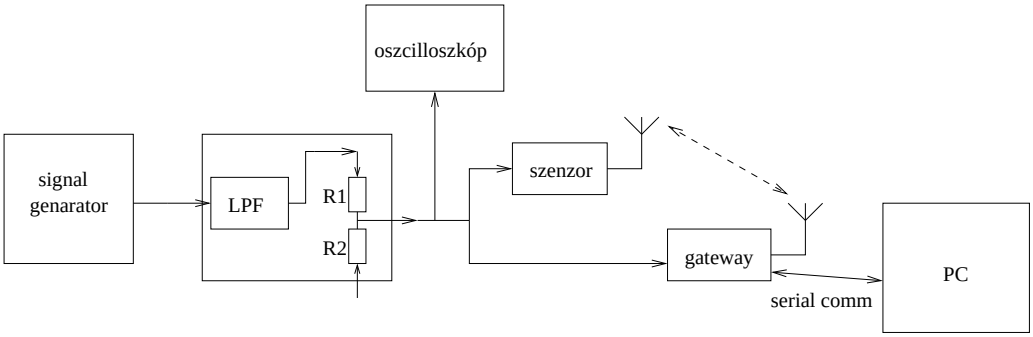
Az interpolációs algoritmus általánosan (a példák alapján) a következő módon hajtandó

végre. Az időzítési diagrammot lásd a 3. ábrán. Tegyük fel, hogy a szenzor által mért értéket a t_p időpontban szeretnénk kiszámítani. A szenzor által egy t időpontban mért értéket jelöljük $s(t)$-vel. Tegyük fel, hogy egy n-ed rendű polinommal szeretnénk az interpolációt elvégezni, ekkor legalább $n+1$ adat szükséges. Egyszerűség kedvéért legyen n páratlan.

1. Keressük meg a t_p előtt közvetlenül beérkező adatot. Az ehhez tartozó időbélyeg: t_r.
2. Az interpolációhoz állítsunk össze legalább $n+1$ elemű vektort a szenzor összetartozó időbélyegeiből és mérési eredményeiből. Legyenek ezek a vektorok rendre: $T = [t_{r-k}, t_{r-k}...t_{r+k}]$ valamint $S = [s(t_{r-k}), s(t_{r-k})...s(t_{r+k})]$, ahol $n \leq 2 \cdot k$.
3. Illesszünk polinomot a T és S pontokra. (Amennyiben szükséges használjunk skálázást és eltolást az időadatoknál, pl. vonjuk ki T-ből t_r-t, és szorozzuk meg valamilyen konstanssal. Ne feledjük ugyanezt elvégezni t_p-re is!)
4. Értékeljük ki az illesztett polinomot a t_p időpontban.

5.2.3. A mérőrendszer felépítése

A mérés első részében a 4. ábrán látható elrendezést használjuk.



4. ábra. A mérőrendszer felépítése az adatgyűjtést megvalósító rendszer esetén (mintavételező és rezonátoros megfigyelőt alkalmazó szenzor esetén is)

Ebben a rendszerben két mitmót található (lásd: 9. ábra). Az egyik pusztán szenzor funkcióval rendelkező node (továbbiakban szenzor). Ez a node az AD átalakítójára vezetett jelet mintavételezi és rádión keresztül továbbítja az átjáróként funkcionáló node felé. Az átjáró (gateway) node fogadja az adatokat, és soros porton továbbítja a PC felé. Az átjáró node ezen kívül szenzor funkciót is ellát, és a szenzorhoz hasonlóan mintavételezi az AD átalakítójára vezetett jelet és a saját adatait is küldi a PC felé soros porton. A mérés első részében ugyanazt a jelet vezetjük a szenzor és az átjáró bemenetére, így ellenőrizhetjük a szinkronizáció pontosságát: mindkét node esetén ugyanazt a jelet kell érzékelni eltekintve az analóg elektronika miatti különbségektől. Külső jelként egy függvénygenerátor jelét használjuk. Ezt a jelet egy egyszerű aluláteresztő szűrőre (LPF: Low Pass Filter) vezetjük, mely sávkorlátozó szűrőként funkcionál, így csökkentve a véges mintavételi frekvencia miatti átlapolódás hatását. (R1 és R2 ellenállások szerepét lásd később, ebben a mérésben nincs hatásuk.) A szűrő a 10. ábrán látható.

Későbbiekben látjuk majd, hogy a szenzor kétféle üzemmódban is képes működni (mintavételezés vagy előfeldolgozás). Az üzemmódot bekapcsolás után SW1 és SW2

nyomógombokkal lépkedve 1-es módba kell állítani (a 7 szegmenses kijelzőn láthatjuk a sorszámot). Választás után az SW3 gomb hosszú megnyomásával nyugtázhatjuk a kiválasztott üzemmódot.

A nodeokon a mintavételi frekvencia névleges értéke megegyezik, mégpedig, 1800.2 Hz, egészen pontosan: $f_s = (8 \cdot 10^6/4444)$ Hz.

Az adatok továbbítása csomag alapú, mind a szenzor mind az átjáró adatai 25 mintát tartalamazó csomagokban kerülnek továbbításra a PC felé. Minden csomag el van látva egy időbélyeggel, ami a csomag első mintájának mintavételezési időpontját mutatja. Ez az időpont az adott node bekapcsolása óta eltelt időt adja meg.

Az átjáró node az adatokon kívül továbbítja a PC felé a szinkronizációs pontokat, melyek segítségével az időbélyeg-transzformációt elvégezhetjük. A szinkronizációs pontok kialakítása a rádiós kommunikáció segítségével történik kihasználva a rádiós IC (CC2420) szolgáltatásait. A CC2420-as IC egy speciális funkciója, hogy adás kezdetekor az adó és vevő oldalon egy digitális kimeneten (SFD láb: Start of Frame Delimiter) egy felfutó élt generál az IC (részletekért lásd az adatlapot). Ez a felfutó él igen nagy pontossággal azonos időpontban generálódik az adó és vevő oldalon, tehát ha mind az adó mind a vevő feljegyzi ezt az időpontot, akkor az így kialakuló időbélygpárok felhasználhatóak szinkronizációs pontokként. Az adott adáshoz tartozó időpontot az adó még az aktuális csomagban elhelyezi, így a vevő a saját maga által feljegyzett időbélyeggel együtt továbbítani tudja a PC felé, ahol ezeket a szinkronizációs pontokat felhasználhatjuk az időbélyeg-transzformáció végrehajtásához.

Az átjáró node által küldött adatokat egy PC-s program segítségével menthetjük el a merevlemezre. Az elmentett file-okat a mérésen rendelkezésre álló `preprocessSampleFiles.m` file meghívásával egyszerűen feldolgozható formátumúra alakíthatjuk.

5. ábra. Az egyszerű mintavételezést alkalmazó mérőrendszer kezelői felülete.

Az előfeldolgozás után a következő file-okat használhatjuk a mérés során:

pre_stmpSync.dat: a szinkronizációs pontpárokat tartalmazó file. A file két oszlopból áll, minden sorban egy-egy szikronizációs pontpár található. Az első oszlopban található az átjáró a második oszlopban a szenzor által generált időpontok, melyek ugyanahhoz a rádiós adáshoz tartoznak. Például egy lehetséges eredmény:

4068.618299375 3.014941125
4068.632187125 3.028829000
4068.646074750 3.042716625
4068.659961625 3.056603625
4068.673849375 3.070491625
4068.687732125 3.084374500
4068.701622875 3.098265250
4068.715510625 3.112153125
4068.729398375 3.126041125

Tehát az első rádiós adás az átjáró bekapcsolása után 4068.618299375 sec-al a szenzor bekapcsolása után pedig 3.014941125 sec-al történt, és ez a két időpont egyazon eseményhez

tartozik.

A file-t (és általában a többit is) a MATLAB load parancsával egyszerűen beolvashatjuk, például:

```
readSync = load('pre_stmpSync.dat');
```

egy két oszlopot tartalmazó vektort ad vissza, mely a file tartalmával egyezik meg.

pre_gwySampFull.dat, **pre_sensorDataFull.dat**: a gateway és a szenzor által küldött adatokat és a hozzájuk tartozó időbélyegeket tartalmazza. A második oszlopban a minták az első oszlopban az adott mintához tartozó időbélyeg található.

5.3. Rezonátor alapú mérőrendszer tanulmányozása

A mérés második részében egy rezonátor alapú tömörítést alkalmazó mérőrendszer vizsgálatára kerül sor. A mérőrendszer fizikai felépítése megegyezik a 4. ábrán látható elrendezéssel, tehát egy szenzor, valamint egy átjáró és szenzor funkciót is ellátó node mintavételezi ugyanazon jelet. A különbség a két rendszerben az, hogy míg az első esetben a szenzor a jel mintavételezését és a minták továbbítását végezte, ezen mérés során a szenzor előállítja a jel Fourier-együtthatóit, és az együtthatókat továbbítja. Ez a rendszer periodikus jelek esetén alkalmazható, mivel a Fourier-együtthatók segítségével periodikus jelek állíthatóak elő. A rendszer előnye, hogy periodikus jelek esetén csökkenthető a rádiós hálózaton átküldendő adatok mennyisége, ugyanis a periodikus jelek Fourier-együtthatói általában lassabban változnak mint maga a jel. Esetünkben a szenzor feleannyi adatot továbbít, mint az egyszerű mintavételező szenzor esetében.

A rendszer demonstrálja az intelligens szenzorok által kínált lehetőséget, nevezetesen a szenzor már helyben alkalmas bizonyos feladatok elvégzésére. Ez a számítási teljesítmény felhasználható a mért jelek előfeldolgozására, mely segíthet a központi egység tehermentesítésében, vagy amint esetünkben is tapasztalható, a kommunikációs teher csökkentésében, hiszen elegendő a mért paramétereket továbbítani. A következő fejezetekben megismerkedhetünk a rezonátoros megfigyelő felépítésével, mely a rendszer egyik alapvető építőeleme.

5.3.1. Rezonátoros megfigyelő áttekintése

A hallgatók a Méréselmélet (vimim108) című tárgy keretei között részletesen megismerkedhetnek a rezonátoros megfigyelő algoritmussal (továbbiakban RBO: Resonator Based Observer). Habár a mérés során nem szükséges az algoritmus részletes ismerete, csupán „alkalmazói" szinten használjuk, egy rövid áttekintést adunk az algoritmusról azért, hogy betekintést kapjunk a rendszer működésébe.

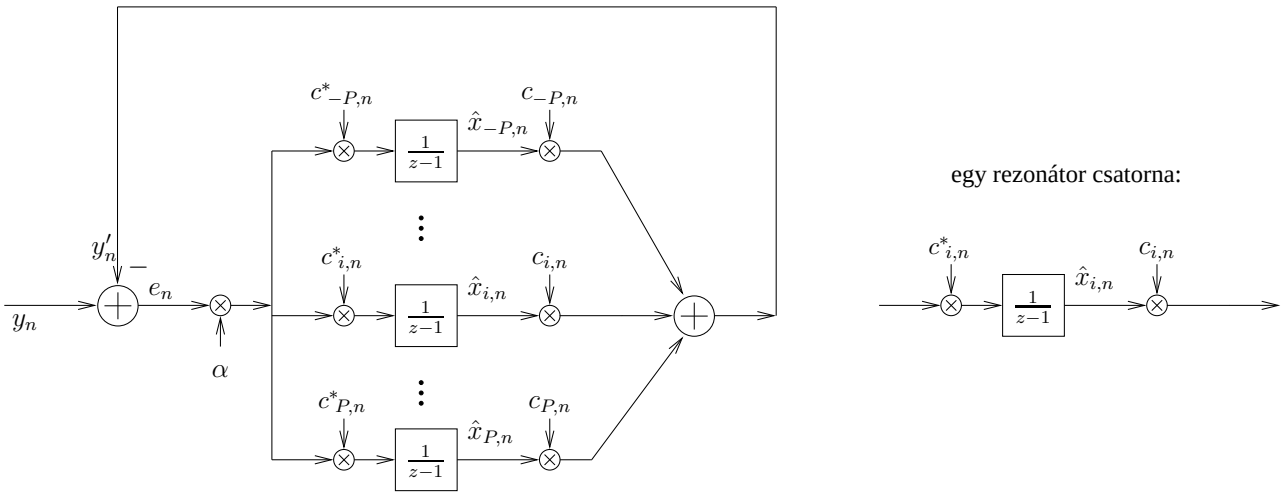
A rezonátor alapú megfigyelő periodikus jelek Fourier-együtthatóinak rekurzív számítására szolgál. Az algoritmus egyik potenciális alternatívája lehet a hagyományos frekvenciatartománybeli diszkrét Fourier-transzformációknak (DFT illetve FFT), azonban a két algoritmus a következő alapvető eltéréseket mutatja:

1. A DFT az egész frekvenciatengely mentén egyenletesen kiszámítja a jel spektrumát, míg az RBO alkalmas csupán a periodikus jelek Fourier-együtthatóinak számítására. Ez jelentős erőforrás megtakarítást jelent, mivel nem kell a spektrumot olyan helyeken is számítani, ahol nem található hasznos komponens.

2. Az RBO-hoz szükséges a jel frekvenciájának ismerete, míg a DFT-nél nem. Amennyiben ismert a jel frekvenciája, az RBO kiküszöböli a DFT esetén fellépő tetőesés (picket fence) és szivárgás (leakage) jelenségeket.

3. A BRO rekurzívan állítja elő a Fourier-együtthatókat, tehát mintáról-mintára frissíti a becslést. Ezzel szemben a DFT blokkos feldolgozást végez, tehát a transzformációhoz szükséges összes adatot össze kell gyűjteni a transzformáció elvégzése előtt, mely adott pontosság esetén általában nagyobb késleltetést visz a rendszerbe.

Az RBO algoritmus megismerése előtt bemutatjuk a periodikus jelek felírásának módját, mely azért szükséges, mivel az RBO periodikus jelek paramétereit méri. Legyen adott egy y_n periodikus jel, mely P számú harmonikust tartalmaz, a mintavételi frekvenciához viszonyított relatív frekvenciája Θ, az i-edik harmonikushoz tartozó Fourier-együtthatói az n időpontban



6. ábra. A rezonátoros megfigyelő blokkvázlata

pedig $x_{i,n}$. Ekkor az y_n jelet a következő alakban írhatjuk fel:

$$y_n = \sum_{i=-P}^{P} x_{i,n} e^{\jmath \Theta i n} = \sum_{i=-P}^{P} x_{i,n} c_{i,n}, \quad \text{és} \quad c_{i,n} = e^{\jmath \Theta i n} \tag{4}$$

ahol n az időindex, $c_{i,n} = e^{\jmath \Theta i n}$ pedig az i-edik felharmonikushoz tartozó bázisfüggvény. A fenti egyenlet általános mintavételes rendszerekben érvényes. Amennyiben valódi időskálát használunk, akkor a következő érvényes:

$$y(t) = \sum_{i=-P}^{P} x_i(t) e^{\jmath f i t} = \sum_{i=-P}^{P} x_i(t) c_i(t). \tag{5}$$

Ebben az esetben f a frekvenciát jelöli, így f_s mintavételi frekvenciát feltételezve: $\Theta = f/f_s$. A kétféle leírási mód –(4) és (5)– azért szükséges, mert a mérés során elosztott működésű rendszerekkel foglalkozunk, ahol általánosságban nem beszélhetünk globális mintavételi időpontokról, viszont az egyes egységeken belül már lokális mintavételi időpontokról igen. Így pl. az egyes egységeken futó algoritmusok leírásánál jobban használható (4), viszont az egymás között cserélt adatok értelmezése csak egy abszolút, közös időskálával lehetséges, mint ahogyan (5) esetén látjuk.

(4) egyenletnek létezik egy tömörebb formája is, mely a felírást egyszerűsíti. Ebben felhasználjuk a következő jelöléseket:

$$\begin{aligned} \mathbf{x}_n &= [x_{-P,n} \quad \dots \quad x_{i,n} \quad \dots \quad x_{P,n}]^{\mathrm{T}} \quad : \text{ oszlopvektor} \\ \mathbf{c}_n &= [c_{-P,n} \quad \dots \quad c_{i,n} \quad \dots \quad c_{P,n}]^{\mathrm{T}} \quad : \text{ oszlopvektor,} \end{aligned} \tag{6}$$

ahol $\mathbf{c}_n$ a bázisfüggvényeket tartalmazó vektor az n-edik időpontban $\mathbf{x}_n$ pedig a Fourier-együtthatókat tartalmazó vektor az n-edik időpontban. Ezekkel a jelölésekkel, felhasználva a vektorszorzás szabályait[1], (4) a következő egyszerű formában írható:

$$y_n = \mathbf{c}_n^{\mathrm{T}} \mathbf{x}_n. \tag{7}$$

[1] $\mathbf{a}^{\mathrm{T}}\mathbf{b} = \sum_i a_i b_i$

Az RBO az y_n jelnek az $\mathbf{x}_n$ Fourier-együtthatóit becsli. Az algoritmus felépítése a 6. ábrán látható. Jelöljük az $\mathbf{x}_n$ Fourier-együtthatók becslőit tartalmazó vektort $\hat{\mathbf{x}}_n$-nel. A megfigyelőt a következő egyenletek írják le:

$$y'_n = \mathbf{c}_n^{\mathrm{T}} \hat{\mathbf{x}}_n, \tag{8}$$

$$e_n = y_n - y'_n = y_n - \mathbf{c}_n^{\mathrm{T}} \hat{\mathbf{x}}_n, \tag{9}$$

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{x}}_n + \alpha e_n \mathbf{c}_n^*. \tag{10}$$

(Megjegyezzük, hogy létezik ettől eltérő felírási mód is. Alkalmazástól függ, hogy melyik felírás praktikusabb, de a felírási mód az alapvető tulajdonságokat nem befolyásolja.) * a komplex konjugálást jelöli. α egy ún. bátorsági tényező, ennek meghatározására nem térünk ki. Kvalitatíve annyi mondható, hogy kis értékű α jó zavarelnyomást eredményez a mérési zajokkal szemben, nagy értékű α gyors beállást biztosít; a konkrét választás kompromisszum eredménye. Látható, hogy az algoritmus követi a rekurzív becslési algoritmusok szokásos alakját:

1. A rendelkezésre álló $\hat{\mathbf{x}}_n$ becsült paraméterek alapján becslést ad az y_n jelre. Lásd (8) egyenletben: y'.
2. Kiszámítja a becslés hibáját: (9).
3. Az e_n becslési hiba alapján módosítja a becsült paramétereket: (10).

Az algoritmus többféleképpen értelmezhető. A következő pontokban felsorolunk néhány értelmezési módot, megkönnyítendő az algoritmus megismerését/megértését:

1. Hagyományos felfogás szerint abból indulunk ki, hogy az y_n jelet úgy tekintjük, mint egy lineáris rendszer kimenetét, melynek állapotegyenletei:

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n \\ y_n &= \mathbf{c}_n^{\mathrm{T}} \mathbf{x}_n. \end{aligned} \tag{11}$$

   Tekinthetjük (11)-t szemléletesen pl. egy függvénygenerátort, mint fizikai rendszert leíró állapotegyenletnek. Ezen lineáris rendszer állapotváltozói a megfigyelt jel Fourier-együtthatói, tehát az ehhez a rendszerhez tervezett állapotmegfigyelő (RBO) a jel Fourier-együtthatóit állítja elő.

2. Az algoritmus formailag nagyban hasonlít az LMS algoritmushoz (lásd Információfeldolgozás laboratórium 3. mérés), és valóban értelmezhető egy komplex értékű LMS algoritmusként. Ebben az esetben $\mathbf{c}_n$ felfogható úgy mint referenciajel, cél pedig, hogy a $\mathbf{c}_n$ bázisfüggvényekkel négyzetes értelemben minél jobban közelítsük az y_n jelet, tehát minimalizáljuk a következő költségfüggvényt: $J(n) = |e_n|^2 = \left|y_n - \mathbf{c}_n^{\mathrm{T}} \hat{\mathbf{x}}_n\right|^2$. $\hat{\mathbf{x}}_n$ paramétervektort minden időpontban a költségfüggvény negatív gradiensének irányába módosítjuk egy α bátorsági paramétert használva. A negatív gradiens (figyelembe véve, hogy $\hat{\mathbf{x}}_n$ komplex szám): $-\nabla J(n) = -\frac{\partial J(n)}{\partial \hat{\mathbf{x}}_n} = \mathbf{c}_n^* \left(y_n - \mathbf{c}_n^{\mathrm{T}} \hat{\mathbf{x}}_n\right) = \mathbf{c}_n^* e_n$. Tehát $\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{x}}_n - \alpha \nabla J(n)$ a (10) egyenletet szolgáltatja.

3. Az egyes rezonátorcsatornák az adott rezonátor frekvencián végtelen átvitellel rendelkeznek, tehát felfoghatóak úgy, mint a rezonátor frekvenciákra eltolt integrátor: a $c_{i,n}^*$-al való szorzás eltolja a i-edik rezonátor frekvencián található jelet DC-re, ahol integráljuk, majd $c_{i,n}$-al való szorzás visszatolja az intergált jelet az eredeti frekvenciára. Szabályozástechnikából ismert, hogy egy visszacsatolt integrátor hiba nélkül tud követni konstans jelet, így a rezonátor frekvenciákra eltolt végtelen erősítés segítségével hiba nélkül előállítható minden, a rezonátor frekvenciákon található periodikus jel is.

A konkrét rendszerben a szenzor maximum 5 darab Fourier-együttható számítására képes. Ez az érték a mikrokontroller (ATmega128 @ 8 MHz) számítási teljesítménye által korlátozott. A node konfigurálható úgy is, hogy csupán a páratlan Fourier-együtthatókat számítsa. Ez azokban az esetekben hasznos, ha a jel csupán páratlan harmonikus pozícióban található Fourier-együtthatókat tartalmaz (pl.: háromszög jel, négyszög jel, vagy általában a villamos hálózat feszültség/áramfüggvénye). Ebben az esetben a páros harmonikusok számítása felesleges számítási teljesítményt kötne le. A DC komponenst a node nem számítja, mivel a nodehoz tartozó interfész kártya AC csatolású. A szenzor nodeot az IO kártyán található 1-es kapcsolóval lehet konfigurálni, hogy az első öt harmonikust (`ON`) vagy az első öt *páratlan* harmonikust számítsa és továbbítsa (`OFF`).

Mivel a megfigyelt y_n jel valós értékű, így teljesül a következő egyenlőség:

$$x_{i,n} = x^*_{-i,n}, \tag{12}$$

tehát a negatív frekvenciákon található Fourier-együttható a pozitív frekvenciákon található komponensek konjugáltjai, emiatt a szenzor csupán $\{x_{i,n};\ i > 0\}$ komponenseket továbbítja, hiszen y_n valós és DC mentes jel egyértelműen előállítható ezen adatokból is:

$$y_n = 2\Re\left\{\sum_{i=1}^{P} x_{i,n}c_{i,n}\right\}, \tag{13}$$

ahol $\Re\{\cdot\}$ a valós rész képzést jelöli. (13) egyszerűen megkapható (4) és (12) egyenletekből, felhasználva, hogy egy γ komplex szám esetén[2]: $\gamma + \gamma^* = 2\Re\{\gamma\}$

Frekvenciamérés rezonátoros megfigyelővel A megfigyelő megfelelő működéséhez meg kell adni a jel frekvenciáját. Erre mérés során egy PC-s kezelői felület szolgál (lásd később a 8. ábrán). A frekvencia pontos mérése a szenzor mérési eredményei alapján történik, mégpedig az Adaptív Fourier Analizátor (AFA) alapgondolatát felhasználva. Ebben a következő összefüggést alkalmazzuk. Legyen adott egy ω_{jel} frekvenciájú és ϕ_{jel} fázisú jel: $y(t) = \cos(\omega_{jel}t + \phi_{jel})$. Amennyiben az $y(t)$ jelet egy ω_1 frekvenciájú bázisfüggvény segítségével állítjuk elő, akkor a következőt kapjuk: $y(t) = \cos[\omega_{jel}t + \phi_{jel}] = \cos\{\omega_1 t + [(\omega_{jel} - \omega_1)t + \phi_{jel}]\}$, amely:

$$y(t) = \cos\{\omega_1 t + [(\omega_{jel} - \omega_1)t + \phi_{jel}]\} = \cos\left\{\omega_1 t + \phi'_{jel}\right\}, \tag{14}$$

tehát egy olyan, ω_1 frekvenciájú jelet kapunk, melynek fázisa a két frekvencia különbségével arányosan változik az időben, tehát:

$$\phi'_{jel} = (\omega_{jel} - \omega_1)t + \phi_{jel} = 2\pi(f_{jel} - f_1)t + \phi_{jel}. \tag{15}$$

Amennyiben a rezonátorok frekvenciáját beállítjuk egy becsült frekvenciára, akkor a jel frekvenciájának a beállított frekvenciától való eltérését az mutatja, hogy a megfigyelt jel fázisa milyen gyorsan változik (ez szemléletesen azt jelenti, hogy a megfigyelt jel Fourier-együtthatója a frekvenciaeltérés arányában forog). Amennyiben ábrázoljuk a fázist, a fázisra illesztett egyenes meredeksége megadja a frekvencieltérést (a meredekség $\frac{\text{rad}}{\text{sec}}$-ben értelmezendő, tehát osztani kell 2π-vel, hogy frekvenciát kapjunk). Egy komplex szám fázisa a MATLAB `angle` parancsával lehetséges, ezen kívül hasznos az `unwrap` parancs használata is (részletekért lásd MATLAB: `help unwrap`).

Manuálisan úgy kereshetjük meg a frekvenciát, hogy az alapharmonikus Fourier-együttható fázisát figyeljük (a rendszerhez tartozó kezelői felületen látható: 8. ábra), és amennyiben az

[2] Amennyiben $\gamma = a + \jmath b$, akkor $\gamma^* = a - \jmath b$, így: $\gamma + \gamma^* = a + \jmath b + a - \jmath b = 2a$, tehát γ valós részének kétszerese.

óramutató járásával ellentétes irányba forog, akkor a frekvenciát növleni kell, ellentétes esetben csökkenteni kell. A forgás sebessége arányos a frekvenciahibával: ha a fázis $d\phi$ fokot változik két egymást követő Fourier-együttható érkezése között, mely $T_F \approx \frac{1}{36}$ sec, akkor a frekvenciaeltérés: $\frac{d\phi}{360° \cdot T_F}$. $d\phi$ értéke közvetlenül leolvasható a rendszerhez tarotzó kezelői felületről: 8. ábrán `dFi`.

5.3.2. Szinkronizáció és jelrekonstrukció rezonátoros megfigyelőt használó rendszerben

A mérés során célunk, hogy a szenzor által előállított, majd az átjáró nodeon keresztül a PC-hez továbbított Fourier-együtthatókból visszaállítsuk a mért jelet. A visszaállított jel összehasonlítható az átjáró node által mintavételezett adatokkal, így tesztelhetjük az algoritmus tulajdonságait.

Fontos megjegyezni, hogy a szenzor node időskáláját ebben a rendszerben is transzformálni kell, ehhez rendelkezésre állnak a szinkronizációs pontok az 5.2.3. fejezetben leírtaknak megfelelő formában.

A szenzor node a Fourier-együtthatókat 50 mintavételi időnként (T_F) továbbítja a PC felé. Mivel egy mintavételi időköz: $T_s = 4444/8 \cdot 10^6 \approx 0.5555$ msec így az együtthatók küldések gyakorisága: $T_F = 50 \cdot 4444/8 \cdot 10^6$ sec $\approx \frac{1}{36}$ sec ≈ 27.8 msec. A Fourier-együtthatókat tartalmazó csomag a következő adatokat tartalmazza:

1. Csomag azonosítója.
2. A csomag küldésének időpontja, a későbbiekben t_r-vel jelöljük.
3. 5 darab Fourier-együttható értéke a csomag küldésének időpontjában.
4. Az alapharmonikus bázisfüggvény fázisa a Fourier-együtthatók küldésének időpontjában, mely: $\varphi_1(t_r)$.

Ez utóbbi azért szükséges, mert egy periodikus jel esetén önmagában egy fázis információ nem mond semmit, meg kell adni egy referenciát. Amennyiben megadjuk, hogy egy adott időpontban alapharmonikus bázisfüggvény fázisa mekkora, akkor a fázis és a bázisfüggvény elvileg bármely más időpontban ezután kiszámítható amennyiben az f frekvencia állandó:

$$\varphi_1(t) = \varphi_1(t_r) + 2\pi f(t - t_r), \tag{16}$$

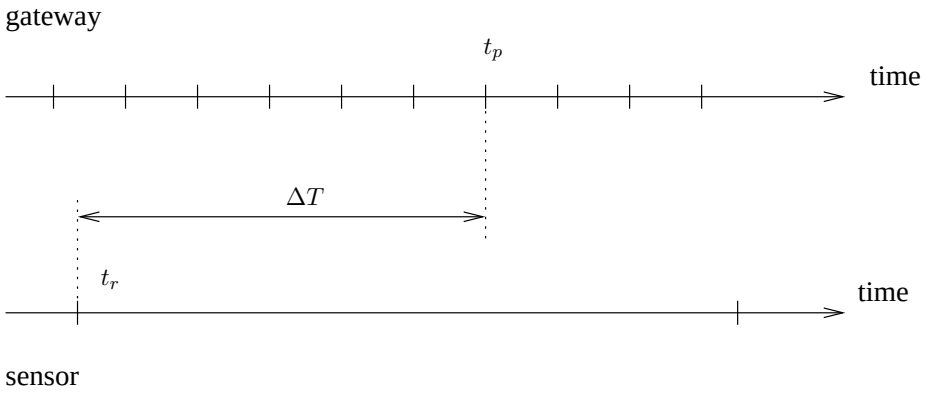
Mivel a frekvencia változhat, ezért szükséges bizonyos gyakorisággal elküldeni a fázist. Az i-edik harmonikus bázisfüggvény fázisa az alapharmonikus fázisának i-szerese:

$$\varphi_i(t) = i\varphi_1(t), \tag{17}$$

a bázisfüggvények pedig:

$$c_i(t) = e^{j\varphi_i(t)}. \tag{18}$$

A mérés során a szenzor és a gateway node ugyanazt a jelet mintavételezi (lásd: 4. ábra). A gateway a nyers mintákat továbbítja, míg a szenzor a Fourier-együtthatókat. A Fourier-együtthatók segítségével a megfigyelt jelet visszaállíthatjuk a t_p időpontban a következő lépések



7. ábra. A Fourier-együtthatókból történő jelrekonstrukció időzítési diagramja

végrehajtásával (időadatok magyarázatához lásd a 7. ábrát):

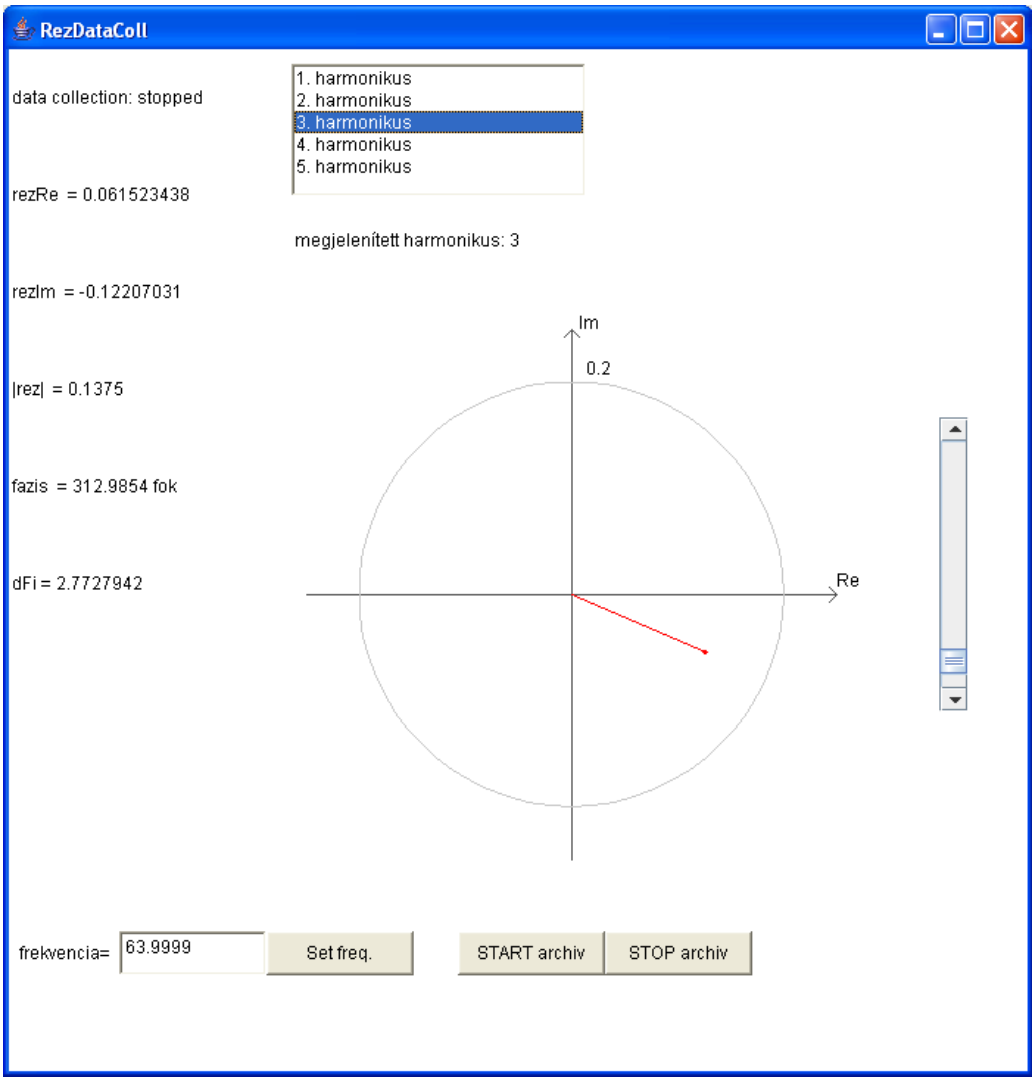
1. Mérjük meg és állítsuk be a jel frekvenciáját, aztán készítsük el a mérési regisztrátumot.
2. Végezzük el a szenzor időbélyegeinek konverzióját.
3. A jel t_p pontban történő előállításához keressük meg a közvetlenül a t_p időpont előtt beérkező legutolsó csomagot. A csomag küldésének időpontja: t_r.
4. Számítsuk ki a csomag érkezése óta eltelt időt: $\Delta T = t_p - t_r$.
5. A csomag küldésekor megadott alapharmonikus fázis értékét ($\varphi_1(t_r)$) módosítsuk a fázis ΔT idő alatt bekövetkező megváltozásával, így megkapjuk az alapharmonikus bázisfüggvény fázisát a t_p időpontban: $\varphi_1(t_p) = \varphi_1(t_r) + 2\pi\Delta T \cdot f$ (lásd: (16)).
6. Számítsuk ki az i-edik harmonikus bázisfüggvény fázisát: $\varphi_i(t_p) = \varphi_1(t_p) \cdot i$ (lásd: (17)).
7. Számítsuk ki a bázisfüggvényeket: $c_i(t_p) = e^{j\varphi_i(t_p)}$ (lásd: (18)).
8. A bázisfüggvények és a szenzor által küldött Fourier-együtthatók (x_i) ismeretében számítsuk ki a jel értékét (13) alapján: $y'(t_p) = 2\Re\left\{\sum_{i=1}^{P} c_i(t_p)x_i\right\}$.

Figyelem! MATLABban a $\sum$ művelet hatékonyan megvalósítható vektorszorzással, lásd pl. (7) egyenletet, melyben az egyik vektor transzponáltját ($^{\mathrm{T}}$) kell számítani. Gyakori hiba, hogy MATLABban a [.'] operátor jelöli a transzpnálást, a ['] operátor a transzponált konjugáltat jelöli, azonban ez valós jelek esetén nem okoz problémát, csak komplex jelek esetén. Figyeljünk arra is, hogy amennyiben `i` vagy `j` változókat ciklusváltozóként használjuk, akkor nem használhatjuk önmagukban a komplex számok képzetes részének jelölésére.

5.3.3. A mérőrendszer felépítése

A rezonátoros megfigyelőt alkalmazó rendszer kezelésére egy PC-s grafikus kezelői felület szolgál, mely a 8. ábrán látható. A kezelői felület a következő lehetőségeket kínálja:

1. Alkalmas a szenzor által számított Fourier-együtthatók megjelenítésére: 8. ábra: pirossal jelölt vektor. A koordináta rendszerben a `Re` tengely a valós az `Im` tengely az együtthatók képzetes részét jelölik. A koordináta rendszer melletti csúszka segítségével (jobb oldal) a



8. ábra. A rezonátoros megfigyelőt alkalmazó mérőrendszer kezelői felülete.

tengelyeket skálázhatjuk. A szürke kör szolgál viszonyítási alapként a vektor hosszának becsléséhez.

2. Az ablak tetején lévő listából kiválaszthatjuk azt, hogy melyik Fourier-együtthatót szeretnénk megjeleníteni.

3. A Fourier-együtthatók az ablak bal oldalán numerikusan is láthatóak algebrai és exponenciális formában: `rezRe` és `rezIm` a Fourier-együttható valós és képzetes részét, a `|rez|` és `fazis` a Fourier-együttható hosszát és fázisszögét adja meg.

4. Szintén az ablak bal oldalán látható `dFi` érték, mely megadja, hogy az alapharmonikus Fourier-együttható fázisa mennyit változott az két csomag küldése között. Ebből megállapíthatjuk, hogy mekkora a frekvenciahiba a megfigyelőnek megadott illetve a jel frekvenciája között, értelmezését lásd korábban a frekvenciaméréshez tartozó leírásban.

5. A `frekvencia=` felirat melletti szövegmezőben megadható a megfigyelő frekvenciája. A beírt értéket `ENTER`-rel vagy a `Set freq.` gomb megnyomásával érvényesítjük. Ekkor a

szenzor nodeon a `LED1` állapota megváltozik.

6. A `START archiv` illetve `STOP archiv` gombokkal a nodeok által küldött adatokat file-ba menthetjük. A `START archiv` gomb megnyomásával elkezdjük a mentést a `STOP archiv` gombbal pedig leállíthatjuk. Minden indítás után a legutóbbi file-okat felülírja a program, de egy külön könyvtárba minden, a mérés során keletkezett file archiválásra kerül.

   A regisztrátumok mentését követően futtassuk a `preprocessResonatorFiles.m` MATLAB scriptet, mely egyszerűen feldolgozható formátumú file-okat generál. A script futtatását követően a következő file-okat használhatjuk.

   signalFreq.dat: a kezelői felületen beállított frekvenciát tartalmazza. Ezen file segítségével a feladatok megoldása során az aktuális méréshez tartozó frekvenciát egyszerűen kiolvashatjuk: `sigFreq = load('signalFreq.dat');`.

   pre_stmpSync.dat, **pre_gwySampFull.dat**: az 5.2.3. szakaszban leírtakkal egyező felépítésű és funkciójú file-ok.

   pre_sensorResonator.dat: a szenzor által küldött adatcsomagokat tartalmazza a következő formában:

   `Msg_ID_1` t_1 $\varphi_1(t_1)$ $\Re\{\hat{x}_1(t_1)\}$ ... $\Re\{\hat{x}_5(t_1)\}$ $\Im\{\hat{x}_1(t_1)\}$ ... $\Im\{\hat{x}_5(t_1)\}$ `rezConf=0`

   `Msg_ID_2` t_2 $\varphi_1(t_2)$ $\Re\{\hat{x}_1(t_2)\}$ ... $\Re\{\hat{x}_5(t_2)\}$ $\Im\{\hat{x}_1(t_2)\}$ ... $\Im\{\hat{x}_5(t_2)\}$ `rezConf=0`

   .

   .

   .

   vagy

   `Msg_ID_1` t_1 $\varphi_1(t_1)$ $\Re\{\hat{x}_1(t_1)\}$ ... $\Re\{\hat{x}_9(t_1)\}$ $\Im\{\hat{x}_1(t_1)\}$ ... $\Im\{\hat{x}_9(t_1)\}$ `rezConf=1`

   `Msg_ID_2` t_2 $\varphi_1(t_2)$ $\Re\{\hat{x}_1(t_2)\}$ ... $\Re\{\hat{x}_9(t_2)\}$ $\Im\{\hat{x}_1(t_2)\}$ ... $\Im\{\hat{x}_9(t_2)\}$ `rezConf=1`

   .

   .

   .

   Tehát az első oszlopban található az adott üzenet azonosítója: `Msg_ID_r`.

   A második oszlopban található az üzenet küldésének időpontja: t_r.

   A harmadik oszlopban található a szenzoron futtatott megfigyelőben az alapharmonikus bázisfüggvény fázisa az üzenet küldésének időpontjában: $\varphi_1(t_r)$.

   Az utolsó (14-edik) oszlopban található `rezConf` flag jelzi, hogy a szenzor az első öt (`rezConf=0`), vagy az első öt *páratlan* Fourier-együtthatót számítja (`rezConf=1`).

   A 4...8 -adik oszlopban található a szenzor által számított Fourier-együtthatók valós része ($\Re\{\hat{x}_i(t_r)\}$). Amennyiben a szenzor az első 5 Fourier-együtthatót számítja (`rezConf=0`), akkor ez az $\left[\Re\{\hat{x}_1(t_r)\},\ \Re\{\hat{x}_2(t_r)\},\ \Re\{\hat{x}_3(t_r)\},\ \Re\{\hat{x}_4(t_r)\},\ \Re\{\hat{x}_5(t_r)\}\right]$ ellenkező esetben (`rezConf=1`): $\left[\Re\{\hat{x}_1(t_r)\},\ \Re\{\hat{x}_3(t_r)\},\ \Re\{\hat{x}_5(t_r)\},\ \Re\{\hat{x}_7(t_r)\},\ \Re\{\hat{x}_9(t_r)\}\right]$

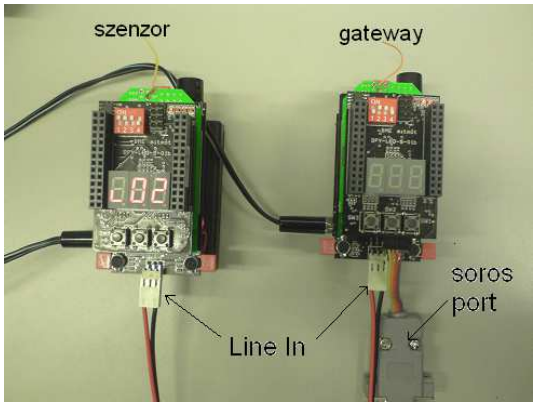
   A 9...13 -adik oszlopban található a szenzor által számított Fourier-együtthatók képzetes része ($\Im\{\hat{x}_i(t_r)\}$). Amennyiben a szenzor az első 5 Fourier-együtthatót számítja

(`rezConf=0`), akkor ez az $\left[\Im\{\hat{x}_1(t_r)\},\ \Im\{\hat{x}_2(t_r)\},\ \Im\{\hat{x}_3(t_r)\},\ \Im\{\hat{x}_4(t_r)\},\ \Im\{\hat{x}_5(t_r)\}\right]$ ellenkező esetben (`rezConf=1`): $\left[\Im\{\hat{x}_1(t_r)\},\ \Im\{\hat{x}_3(t_r)\},\ \Im\{\hat{x}_5(t_r)\},\ \Im\{\hat{x}_7(t_r)\},\ \Im\{\hat{x}_9(t_r)\}\right]$
Ezen adatok alapján a Fourier-együtthatók előállíthatóak: $\hat{x}_i(t_r) = \Re\{\hat{x}_i(t_r)\} + \jmath\Im\{\hat{x}_i(t_r)\}$.

5.4. Felhasznált eszközök összefoglalása

Ebben a fejezetben összegezzük a mérés során felhasznált eszközök bemutatását.

Mitmótok A rendszerben két darab mitmótot alkalmazunk; egyik szenzorként funkcionál a másik pedig bázisállomásként és érzékelőként is funkcionál. Mindkét node rendelkezik egy IO kártyával (DPY-LED-S-01b), egy akusztikus szenzorkártyával és egy 2.4 GHz-es ZigBee rádiós kártyával. Az akusztikus szenzorkártya képes akusztikus jelek érzékelésére valamint található rajta közvetlen, erősítés nélküli vonalbemenet, melyre tetszőleges analóg jelet vezethetünk $0\ V_{pp} - 3.3\ V_{pp}$ tartományban. A kártya AC csatolású, körülbelül 20 Hz-nél található az alsó töréspontí frekvencia.



9. ábra. A mérésben használt mitmótok felépítése

A nodeokon található kapcsolók és nyomógombok segítségével konfigurálhatóak működés közben. A DIP kapcsolón található 4-es kapcsoló segítségével választhatunk, hogy a nodeok a mikrofon vagy a line-in bemenetet mintavételezzék. A mintavételezés frekvenciája $f_s = 8 \cdot 10^6/4444 \approx 1800.2$ Hz.

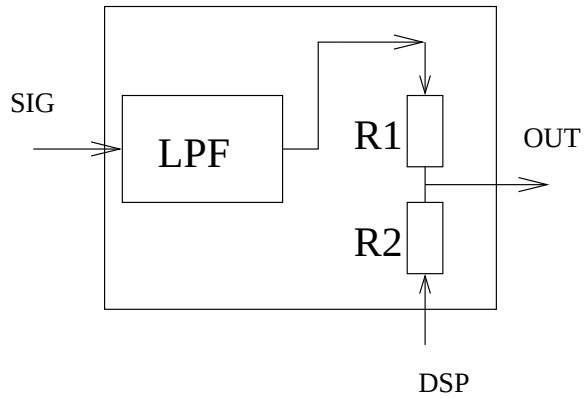
A szenzor node kétféle üzemmódban képes üzemelni. Az üzemmódot a node bekapcsolásakor (vagy reset-et követően) lehet beállítani az `SW1` illetve `SW2` gombokkal lépkedve az üzemmódok között. Az `SW3` gombbal nyugtázhatjuk az üzemmódot. 1-es üzemmódban a szenzor csupán a mintákat továbbítja a gateway felé, 2-es üzemmódban a Fourier-együtthatókat számítja.

Amennyiben a szenzor node a Fourier-együtthatókat számítja, akkor az 1-es kapcsoló segítségével választhatjuk ki, hogy az [1...5] harmonikusokat számítsa, vagy az [1, 3, 5, 7, 9] harmonikusokat.

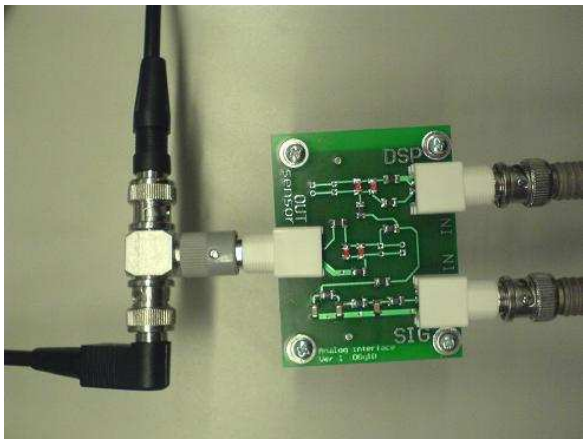
A nodeokon található RESET gomb segítségével újraindíthatóak a nodeok. A RESET gomb akkor használandó, ha új üzemmódban szeretnénk indítani a rendszert vagy rendellenes működést tapasztalunk.

A szenzor nodeon a LED1 állapotváltozása jelzi egy új frekvencia beállítását. A frekvencia a 8. ábrán látható kezelői felület segítségével állítható be.

Analóg interface kártya Az analóg interface kártya (10. ábra) az egyes egységek tehát a jelgenerátor, a DSP és a nodeok közötti összeköttetést teszi lehetővé az analóg jelek számára. A jelgenerátor felől tartalmaz egy egyszerű RC aluláteresztő szűrőt (LPF) a jel sávkorlátozásához.



(a) Analóg interface kártya blokkvázlata



(b) Analóg interface kártya fizikai kialakítása

10. ábra. Analóg interface kártya

5.5. Mérési feladatok

A mérési elrendezések összeállításakor ügyeljünk arra, hogy először tápfeszültséget csatlakoztassuk az eszközökhöz, és csak aztán kössük be a jel illetve adatvezetékeket. Minden esetben ellenőrizzük a nodeokon a kapcsolók állását, mert azok alapvetően befolyásolják a működést.

1. Mintavételező rendszer vizsgálata

   1.1. Állítsd össze a 4. ábrán látható rendszert! A szenzor nodeot 1-es üzemmódban kell elindítani. A szenzoron a DIP kapcsolók állása 1...4 -ig a következő: [ON ON ON OFF]. A gateway-en a DIP kapcsolók állása 1...4 -ig következő: [ON ON ON OFF]. A függvénygenerátoron érdemes először 40 Hz és 100 Hz közötti jelet beállítani, mely elősegíti a jel vizuális kiértékelését.

   1.2. Készíts mérési regisztrátumot! A regisztrátum hossza az első próbák esetén ne legyen néhány sec-nál hosszabb, mert a nagy mennyiségű adat feldolgozása időigényes lehet. A mérés befejeztét követően futtassuk le a `preprocessSampleFiles.m` MATLAB scriptet, mely egyszerűen feldolgozható formátumúvá konvertálja mentett adatokat. Továbbiakban a 'pre_' előtagú file-okat kell feldolgozni: `pre_stmpSync.dat`, `pre_gwySampFull.dat`, `pre_sensorDataFull.dat` (lásd: a 10. oldalon)
   A feladatokat végezd majd el egy 50 Hz-es és 500 Hz-es jelekkel is úgy, hogy mérés közben legalább egyszer ki és bekapcsolod a jelgenerátort. A jel amplitúdója lehetőleg töltse ki a mérési tartományt, de kerüljük a túlvezérlést, legyen némi tartalék a rendszerben. Az eredmények akkor a leginformatívabbak, ha a mérés során megvárjuk, hogy a szenzor és bázisállomás minél inkább „szétcsússzanak“ egymáshoz képest. Ez akkor következik be, ha az adatgyűjtő programban a 'mintaveteli kulonbseg' megjelenített értéke 2000 vagy -2000 körül jár. A dat fájlokat archiváld manuálisan!

   1.3. Vizsgáld meg a szinkronizációs pontokat a `pre_stmpSync.dat` file alapján, és ábrázold a két node lokális idejét egymáshoz képest! Határozd meg az időbélyegtranszformációhoz szükséges paramétereket (lásd: (2)), majd végezd el az időbélyegtranszformációt a meghatározott paraméterek alapján az 5. oldalon leírt módszer felhasználásával! Ábrázold a regisztrátumok időfüggvényeit az időbélyeg-transzformáció előtt és után, és értékeld az eredményeket! Készíts mérési regisztrátumot miközben folyamatosan változtatod a jel amplitúdóját!

1.4. Az időbélyeg-transzformáció megvalósítását végezd el „manuálisan“ felírva LS becslés segítésével! (Építsd fel az $\mathbf{X}$, $\mathbf{y}$ mátrixot és vektort, majd a $\mathbf{p} = \mathbf{X}^T(\mathbf{X}^T \cdot \mathbf{X})^{-1}$ képlettel határozd meg a szükséges paramétert.) Told el a nullától a szenzor szinkronizációs időpontjait 10^6 sec-mal. Hogyan változik a feladat kondicionáltsága? Mi a megoldás, ha ennyire különbözik a két egység ofszetértéke?

1.5. Számítd ki a szenzor mérési eredményeit a gateway mintavételi időpontjaiban lineáris interpoláció (lásd: (3)) segítségével (lásd MATLAB `interp1`)! Hasonlítd össze a két node által mért adatokat! Növeld meg a jel frekvenciáját és végezd el a mérést így is! Összehasonlítási alapként használható a négyzetes eltérés mérése.

1.6. Végezd el a szenzor mérési eredményeinek egészszámszoros interpolálását (beleértve az interpolálás miatt megjelenő új mintákhoz az időbélyeg generálását), és ezeken az adatokon végezd el a lineáris interpolációt! Javít-e a módszer nagy frekvenciás jelek esetén az interpoláció pontosságán? Tervezd meg az interpoláló szűrőt 10-szeres interpoláláshoz. Szűréshez használd a `filtfilt` parancsot a késleltetések kiküszöbölése céljából!

1.7. Számítsd ki a szenzor mérési eredményeit a gateway mintavételi időpontjaiban magasabb rendű interpoláció segítségével (különböző fokszámokkal lehet próbálkozni)! Hasonlítd össze a két node által mért adatokat! Növeld meg a jel frekvenciáját és végezd el a mérést így is! Összehasonlítási alapként használható a négyzetes eltérés mérése. Interpolációra használható a MATLAB `polyfit / polyval` függvénypárosa. Ezek a függvények nem a teljes adatsoron végeznek interpolációt, hanem mintáról-mintára haladva kell ezt elvégezni: a gateway mintavételi időpontjain (t_i) lépkedve keressük meg az adott mintavételi időponthoz legközelebb eső szenzor időpontot (n_0 index), majd az $[n_0 - L \ldots n_0 + L]$ indexű elemekre illesszünk $2 \cdot L$ fokszámú polinomot, és azt értékeljük ki a t_i időpontban.

2. Rezonátoros adatgyűjtő rendszer vizsgálata

2.1. A mérési összeállítást változatlanul hagyva resetelje a nodeokat! A szenzor nodeot 2-es üzemmódban kell elindítani. Ebben az esetben a szenzor a Fourier-együtthatókat továbbítja a PC felé. A szenzoron a DIP kapcsolók állása 1...4 -ig a következő: [ON ON ON OFF]. A gateway-en a DIP kapcsolók állása 1...4 -ig következő: [ON ON ON OFF]. A függvénygenerátoron érdemes először 40 Hz és 100 Hz közötti jelet beállítani, mely elősegíti a jel vizuális kiértékelését. Az adatgyűjtést befejezve a `preprocessResonatorFiles.m` script meghívásával generálhatjuk az egyszerűen feldolgozható formátumúvá konvertált file-okat: `pre_stmpSync.dat, pre_gwySampFull.dat, pre_sensorResonator.dat, signalFreq.dat` (lásd: a 18. oldalon)

2.2. A PC-s kezelői felületen állítson be egy becsült frekvenciát (pl. oszcilloszkópon mérve a periódusidőt)! Mérje meg pontosan a jel frekvenciáját az alapharmonikushoz tartozó Fourier-együttható segítségével a következő módon:

a) Az alapharmonikus Fourier-együttható forgási irányát figyelve növelje illetve csökkentse a frekvenciát, míg a Fourier-együtthatót reprezentáló vektor forgása megáll.

b) Az alapharmonikus Fourier-együttható forgásának periodusidejét mérve (manuálisan). (Az idő mérésére használhatja pl. a MATLAB `tic` és `toc` függvényeit).

c) Az alapharmonikus Fourier-együttható fázisát (lásd: `anlgle` és `unwrap` függvények) ábrázolja és illesszen egyenest a fázis időfüggvényére. (15)-t felhasználva az egyenes meredeksége a körfrekvencia-eltéréssel egyezik meg.

2.3. Készíts mérési regisztrátumot! A regisztrátum hossza az első próbák esetén ne legyen néhány sec-nál hosszabb, mert a nagy mennyiségű adat feldolgozása időigényes lehet. A gateway adataiból távolítd el a DC komponenst (vond ki az átlagot), mert az csupán az unipoláris ADC miatt van a jelben, a rezonátoros megfigyelő a DC komponenst viszont nem számítja. Az előzőekben leírtak alapján végezzd el az időbélyegtranszformációt és állítsd elő a megfigyelt jelet a Fourier-együtthatókból a gateway mintavételi időpontjaiban! A rekonstrukció algoritmusa a 16. oldalon olvasható. A rekonstrukcióhoz szükséges a jel frekvenciájának ismerete, ezt a `signalFreq.dat` fileból olvashatjuk ki. Hasonlítd össze a mintavételezett és a Fourier-együtthatókból rekonstruált jelet! Hasonlítd össze a visszaállított és mintavételezett jelet különböző frekvenciájú és változó amplitúdójú jelek esetén is! Hogyan tudja a rendszer követni az amplitúdóváltozást? Rekonstrukció során érdemes a $\mathbf{c}_n$ bázisfüggvények időfüggvényeit kirajzolni (komplex bázisfüggvény valós és képzetes részét vagy a trajektóriát a `plot3` függvénnyel).

2.4. Vizsgáld meg, hogy hogyan változik a becsült amplitúdó, amennyiben a valóditól eltérő frekvenciát állítunk be, pl. ± 0.01 Hz, ± 0.1 Hz, ± 0.5 Hz, ± 1.0 Hz, ± 2.0 Hz, ± 5.0 Hz, ± 10.0 Hz ... Mérd meg az algoritmus átviteli függvényét és ábrázold! (legyen az alapharmonikus frekvencia 50 Hz) Hogyan viszonyul az eredmény a DFT esetén megismert átviteli függvényhez?

2.5. *Kiegészítő feladat*: Vizsgáld meg, hogy különböző frekvenciabecslési hibáknál milyen pontosan lehet visszaállítani az időtartománybeli jelet! (Frekvenciabecslési hiba alatt azt értjük, hogy mennyivel tér el a jel valódi frekvenciája a beállított értéktől.)

2.6. A rezonátoros megfigyelőt alkalmazó rendszerben a gateway adatai alapján számítd ki a jel Fourier-együtthatóit és hasonlítsd össze a szenzor által szolgáltatott adatokkal! Lásd: (8)-(9)-(10) egyenletek.

2.7. *Kiegészítő feladat*: Vizsgáld meg, hogy milyen hibával állítható vissza egy mért jel, amennyiben a szenzor az első öt, illetve az első öt *páratlan* Fourier-együtthatót méri! Vizsgálójelként használjon szinusz, háromszög illetve négyszög jelet! Azt, hogy a szenzor node melyik együtthatókat számítsa a szenzor nodeon található 4-es kapcsolóval lehetséges kiválasztani (`ON`: 1...5, `OFF`: 1,3,5,7,9). Ezt a mentett file utolsó oszlopa is jelzi, ahogyan az bemutatásra került a 18. oldalon.

2.8. *Kiegészítő feladat*: A rezonátoros adatgyűjtő rendszer esetén töröljük a szenzor adatainak bizonyos százalékát, ezzel szimulálva azt, hogy a szenzor ritkábban továbbítja az adatokat! Vizsgáljuk meg azt, hogy milyen hatása van a jelrekonstrukcióra az egyre ritkuló mérés (tranziens és állandósult állapotban is)!