Опис Герцеловог алгоритма за препознавање тонова на ДСП-у

Јун 2004.

Contents

1 Увод

Овај напис не садржи ништа посебно ново, али садржи све оно што по правилу не пише у неким књигама, чланцима и сличном. Описује се примена Герцеловог алгоритма за препознавање тонова. У извесном смислу, овај напис је подсетник "шта ту како иде" када се хоће да примени Герцелов алгоритам.

Нису разматране алтернативе, има их много. Нису даване ни широке теоријске основе, јер тога може да се нађе на много места. Дате су само најважније дефиниције и теоријске основе неопходне за разумевање остатка написа, који већ не може да се нађе на много места.

1.1 Неке теоријске основе

У наставку дајемо неке теоријске основе из области телекомуникација и обраде сигнала.

Ниво сигнала у "миливатним" децибелима (дБм) се рачуна као:

$$P_{dBm} = 10 \log(\frac{P}{P_{mW}}) \tag{1}$$

Где је $P_{mW} = 1mW$, а P је снага у Ватима. Подразумева се логаритам за основу 10. Из ове једначине се добија једначина за снагу, на основу нивоа у децибелима:

$$P = P_{mW} \cdot 10^{\frac{P_{dBm}}{10}} \tag{2}$$

Снага у зависности од напона се рачуна на отпорнику од $R_{ref} = 600\Omega$:

$$P = \frac{V_{eff}^2}{R_{ref}} \tag{3}$$

Одатле је ефективан напон:

$$V_{eff} = \sqrt{R_{ref} P} \tag{4}$$

А одатле је амплитуда напона:

$$V_m = V_{eff}\sqrt{2} = \sqrt{2R_{ref}P} \tag{5}$$

Нумерички може да се добије једноставна формула (у СИ јединицама):

$$V_m = \sqrt{1200P} \tag{6}$$

Одатле може да се дође до податка колики је ниво сигнала ако му је амплитуда одређене вредности, по формули:

$$P_{dBm} = 10 \log \frac{V_{eff}^2}{R_{ref} P_{mW}} = 10 \log \frac{V_m^2}{2R_{ref} P_{mW}} \tag{7}$$

Односно, чисто нумерички:

$$P_{dBm} = 10 \log \frac{V_m^2}{1,2} \tag{8}$$

1.2 А закон

У Телефонији (Европској и сродној) се говор преноси спакован по А Закону. А Закон је тако направљен да се говор преноси без много изобличења (требало би да је прилагођен начину на које људско ухо прима звук), али истог наравно има (чувено "другачије ми звучиш преко телефона"). У Јапану и Америци се користи μ Закон, који је сличан.

А Закон врши паковање/распаковање означеног податка од дванаест бита у осам бита. Добијени бајт можемо да разложимо овако (једно слово представља један бит):

$$\underbrace{S}_{znak}\ \underbrace{XXXX}_{mantisa}\ \underbrace{YYY}_{eksponent}$$

Практично, А закон је нека врста записа у покретном зарезу. Очигледно, А закон "чува" само четири бита корисних података о одбирку "самом по себи", али чува и знак и експонент. Декодовање по А закону може да изврши следеће парче Ц кода:

```
uint16_t Dekodiraj_A_Zakon(uint8_t byOdbirak)
{
    uint16_t wZnak = (byOdbirak & 0x80) << 5;
    uint16_t wMantisa = byOdbirak & 0x78;
    int eksponent = byOdbirak & 0x07;
```

```
    return wZnak | (wMantisa << eksponent);
}
```

За љубитеље Ц-овске концизности и брзине:

```
inline uint16_t Dekodiraj_A_Zakon(uint8_t by) {
    return ((by & 0x80) << 5) | ((by & 0x78) << (by & 0x07));
}
```

Пошто се ради само о 256 могућности, обично се ово сачува у табели. Такође, зависно од потреба, може знак да се "прошири до краја", односно да добијена реч (два бајта) такође буду означене. У Волтима, А Закон може да представи вредности волта од приближно $\pm 1,5V$. Давање пуне табеле А Закона овде би било некорисно, али, следеће може да буде занимљиво:

- У распакованом облику, један Волт приближно представља цео број 2619, односно 0xA3b.
- Ако се распаковани облик тумачи као број у фиксном зарезу, опсега од -1 до 1 (тако се често ради на ДСП-овима), онда 2619, односно један Волт, износи приближно: 0,07992554, односно коефицијент скалирања Волт/фиксни зарез износи 12.5144596.

Кодирање је нешто теже (између осталог, треба доћи до 12 бита које треба кодирати), али се у применама које нас занимају практично никад не користи. За свирање тонова се обично направи табела одбирака (осмобитних) које треба послати без обраде.

1.3 Рачунања у дигиталној обради сигнала (целобројна, фиксни зарез...)

Главни "штос" у дигиталној обради сигнала је одрадити посао са неприлагођеним алатима, а да би исти коштали мање и да би могли што више канала да обраде. Наиме, ако знамо формуле и имамо процесор који уме да рачуна у покретном зарезу од 80 бита (у Ц-у је то `long double` тип), онда само треба да га пустимо да одради свој посао, и, за све употребе у нама важне сврхе, неће бити никаквих грешака, испадања из опсега и слично. Чак је и 32-бита (у Ц-у `float` тип) сасвим довољно и они скупљи ДСП-ови управо раде са тим типом података. Међутим, у већини случајева се не користе ти скупљи ДСП-ови.

У пракси (а посебно у ГВС-у) се користе 16-битни ДСП-ови са "непокретним зарезом". У ГВС-у је највише коришћена фамилија МЦ56ххх. Основна одлика овог процесора је што има 16-битну архитектуру, са два акумулатора од 40 бита (на 5616х, а 36 бита на 5685х). Подршку за непокретни зарез, је у опсегу -1 до 1 (прецизније -1 до $1 - 2^{15}$). Практично, целобројни садржај неког регистра треба да поделимо са 2^{15} и добићемо вредност у опсегу -1 до 1. Ево и неколико примера:

Хекса	Бинарно	Декадно	Бинарно (зарез)	фиксни зарез
7FFF	0111 1111 1111 1111.	32767	0.111 1111 1111 1111	0.99997
7000	0111 0000 0000 0000.	28672	0.111 0000 0000 0000	0.875
4000	0100 0000 0000 0000.	16384	0.100 0000 0000 0000	0.5
2000	0010 0000 0000 0000.	8192	0.010 0000 0000 0000	0.25
1000	0001 0000 0000 0000.	4096	0.001 0000 0000 0000	0.125
0000	0000 0000 0000 0000.	0	0.000 0000 0000 0000	0.0
C000	1100 0000 0000 0000.	–16384	1.100 0000 0000 0000	–0.5
E000	1110 0000 0000 0000.	–8192	1.110 0000 0000 0000	–0.25
F000	1111 0000 0000 0000.	–4096	1.111 0000 0000 0000	–0.125
9000	1001 0000 0000 0000.	–28672	1.001 0000 0000 0000	–0.875
8000	1000 0000 0000 0000.	–32768	1.000 0000 0000 0000	–1.0

У већини алгоритама се користе означени бројеви. Знак носи један бит (највиши). Бројеви се представљају у потпуном комплементу. Такође, најчешће се користе наредбе за рад у непокретном зарезу. Шта то, у ствари значи? Постоји само пар ситних разлика. Пре свега, сабирање и одузимање је исто (не постоје ни другачије наредба!). Разлике су у множењу и дељењу. Дељење се ретко користи у оваквим алгоритмима, па га прескачемо.

Што се множења тиче, у оба случаја се изврши множење два означена броја. Пошто се множе два 16-битна означена броја, резултат је означен 31-битан број (у општем случају, множење два N-то битна броја означена броја даје 2*N-1 - битан означен број). Питање је шта са тим бројем урадити до пуних 32 бита? Одговор:

- Ако је цео, проширити до 32 бита, са чувањем знака
- Ако је у фиксном зарезу, померити улево за један (као множење са два)

На тај начин, 32 битни број остаје у истом опсегу као и полазни 16-битни бројеви. За даљи рачун се, по правилу, узимају највиших 16 бита, при чему то и није баш јасно шта тачно значи ако причамо у целим бројевима. У фиксном зарезу, просто смо одбацили мање важне битове.

Треба обратити пажњу на прекорачења опсега. Наиме, множење два броја између -1 и 1 неће изаћи из опсега, али, сабирање можда хоће. Дакле, у току рада, може се изаћи из опсега и о томе мора да се води рачуна. Различити алгоритми се према овоме различито односе, некима је свеједно, али, најчешће је то ознака грешке и потребе да се рачун прогласи неисправним. Ако се ради о препознавању тонова, онда то углавном значи да треба одустати од препознавања - било да се тај рачун просто заборави, или пријави грешка да би било обрађено (пријављено, забележено).

У вези са претходним - често се дешава да у формулама за рачунање у обради сигнала стоје неке целобројне константе - два или тако нешто. Њих не треба преводити у непокретни зарез (узгред, то не може тривијално да се изведе, јер би требало и штошта друго онда на неки начин скалирати, који може да буде врло нелинеаран). Ако се ради о броју 2, али и ако се ради о било ком степену двојке, онда овде треба просто применити померање за одговарајући број бита улево. Уколико се деси да тако померен број оде ван опсега, онда се поново ради о нерегуларном случају који смо малопре поменули (све ће бити јасније ако се присетимо да је: $2x = x + x$ - чак можемо "намерно" и да применимо сабирање са самим собом уместо померања, ако се ради о броју 2).

Још једна ствар у вези са прекорачењем опсега. Уколико нам се то често дешава, а прецизност нам није важна, или нам просто "мали сигнали" нису важни, онда можемо да избацимо неколико бита из улазног сигнала. Рецимо, ако је то улаз после А Закона, онда можемо, уместо 12 да користимо, рецимо, 10 бита. Наравно, раније поменуте односе Волта и 12-битних целобројних (или непокретно-зарезних) вредности треба са тиме ускладити (у датом случају, помножити или поделити са 4).

2 Герцелов алгоритам

Герцелов алгоритам је алгоритам прорачуна једног резултата Дискретне Фуријеове Трансформације (скр. ДФТ). Дакле, прорачун једног "фазног одбирка" ДФТ-а.

Главна предност у односу на "пуни" ДФТ је што Герцелов алгоритам, практично, представља филтер, односно врши се за неку одређену (мање-више произвољну) учестаност, док се ДФТ врши на више учестаности, равномерно распоређених по опсегу разматраних учестаности. Ово је врло корисно за, рецимо, препознавање ДТМФ тонова, пошто су они

на “чудним учестаностима”, што значи да је готово неизводљиво спровести ДФТ над њима, већ је могуће извести ДФТ на неким блиским учестаностима.

Такође, Герцелов алгоритам се изражава у рекурзивној формули (представља филтер бесконачног одзива другог реда), што може да буде лакше за кодирање или брже за рачунање на неком процесору, или у неком окружењу (рецимо, зависно од тога како нам стижу одбирци).

Овде нећемо изводити цео алгоритам, већ само дајемо “коначно решење”.

2.1 Дефиниција

Ако са $x(n)$ означимо један одбирак (н-ти) у низу, а са $V(n)$ н-ти члан Герцеловог низа, и са K_k Герцелов коефицијент за учестаност на којој се алгоритам спроводи (или врши филтрирање), онда важи рекурзивна формула:

$$V(n) = K_k V(n-1) - V(n-2) + x(n) \tag{9}$$

При чему се узима да је:

$$V(-1) = V(-2) = 0$$

Герцелов коефицијент за дату учестаност f је:

$$K_k = 2\cos 2\pi \frac{f}{f_s} \tag{10}$$

Где је f_s учестаност одабирања, која у Телефонским применама износи:

$$f_s = 8kHz$$

Одавде може да се закључи да је за Герцелов алгоритам могуће чувати само две променљиве, није потребно чувати цео низ. Такође, постоји само један коефицијент који се увек примењује. Ако се ради “у цугу”, онда за рачун може да важи следећи Руби код:

```
v_n = v_n_1 = 0
0.upto(N) { |n|
    v_n, v_n_1 = Kk*v_n - v_n_1 + x[n], v_n
}
```

Намерно је изабран Руби, да не би морали да одређујемо тип променљиве и да се бакћемо са опсезима вредности.

На крају, односно, након што обрадимо свих N одбирака, од тако добијених $V(n)$ и $V(n-1)$, добијемо следећи израз за квадрат амплитуде обрађеног сигнала на датој учестаности:

$$V_{mG}^2 = V(n)^2 + V(n-1)^2 - 2\cos(2\pi\frac{f}{f_s})V(n)V(n-1) \tag{11}$$

Овде је намерно дат пун израз, јер даје извесан наговештај о својој природи (ко је први рекао “Косинусна теорема”?), а можемо да уочимо да се опет користи Герцелов коефицијент, односно да ово може краће да се напише:

$$V_{mG}^2 = V(n)^2 + V(n-1)^2 - K_k V(n)V(n-1) \tag{12}$$

Овај резултат је у "квадратима Волта", али, у *фреквентном домену*, а не у *временском домену*. То звучи много паметно, али, у пракси, ради се само о томе да V_{mG}^2 није скалирана вредност. Да би је превели у *временски домен*, треба извршити:

$$V_m^2 = (\frac{V_{mG}}{\frac{N}{2}})^2 = \frac{4 \cdot V_{mG}^2}{N^2} \tag{13}$$

За објашњења зашто тако, погледати неку књигу са описом ДФТ.

Пошто се углавном о сигналу говори у децибелима ("миливатним"):

$$P_{dBm} = 10 \log \frac{4V_{mG}^2}{2R_{ref}P_{mW}N^2} = 10 \log 2 \frac{V_{mG}^2}{R_{ref}P_{mW}N^2} \tag{14}$$

Како је ово релативно незгодан рачун за један ДСП (логаритам се рачуна нумерички), онда се често унапред одреде нивои у децибелима, а у обради користе претворени у квадрате Волта, за шта важи формула:

$$V_{mG}^2 = \frac{R_{ref}P_{mW}N^2}{2} 10^{\frac{P_{dBm}}{10}} \tag{15}$$

Односно чисто нумерички:

$$V_{mG}^2 = 0,3N^2 \cdot 10^{\frac{P_{dBm}}{10}} \tag{16}$$

Ако Герцелов алгоритам посматрамо као филтер, онда за њега важи да му је пропусни опсег:

$$B = \frac{f_s}{N} \tag{17}$$

пошто је то ширина једне "фреквентне канте" у ДФТ. Како нам је пропусни опсег често унапред задат, онда на основу тога можемо да одредимо број одбирака:

$$N = [\frac{f_s}{B}] \tag{18}$$

Ако нема неких других захтева, ово је често добар број. Међутим, обично се умешају разни захтеви, пре свега брзина препознавања, али и опсези препознавања и непрепознавања, па није тако лако доћи до броја одбирака.

Ипак, ваља приметити да број одбирака не може да буде баш произвољан, бар не без неких последица. Наиме, као што рекосмо, Герцел је просто алгоритам за прорачун једног резултата ДФТ-а. Дакле, N треба наместити тако да један резултат ДФТ-а (онај за који вршимо рачун Герцеловим алгоритмом) буде такав да нашу учестаност ухвати на средини своје фреквентнне канте, односно, тако да се смањи количина расипања те наше учестаности на суседне "канте" при извођењу ДФТ-а, односно Герцеловог алгоритма.

2.2 Запажања

Ако се гледа како Герцелов алгоритам "ради", види се да $V(n)$ добијају "разне" вредности, као да осцилују. С друге стране, $V_{mG}^2(n)$ монотоно расте. У принципу, требало би да $V_{mG}^2(n)$ може математички да се тумачи као низ који ковергира ка квадрату амплитуде датог сигнала на датој (Герцеловој) учестаности.

Мало пробања Герцеловог алгоритма у акцији (над одбирцима А закона) показује да вредности које добија $V(n)$ могу да буду и преко 300 (у СИ јединицама, дакле, "некаквим" Волтима). У односу на опсег А-закона, то је преко 256 пута веће. Дакле, у тим неким

најгорим случајевима, потребно је и 8-9 битова више него што их има у А-закону, па, за потпуно математички чист прорачун испада да нам треба бар 21-битни процесор, односно 24-битни, пошто се 21-битни углавном не праве. Посебно што добијена снага иде и преко 20000, односно за њу нам, опет за чист рачун, није доста ни 24 бита.

Због тога се често примењује слабљење сигнала, као што је већ поменуто, померањем резултата А-закона за неколико бита (два је некако најчешћи случај - померање за два бита је дељење са четири). Ово слабљење зовемо и "глупим" слабљењем, јер се спроводи увек, без анализе добијеног сигнала.

Треба приметити да вредности за $V(n)$ расту са опадањем Герцелове учестаности, што није баш интуитивно, јер $V(n)$ зависи од Герцеловог коефицијента K_k. За већину примена, тражи се учестаност до 3000 херца, што нас, по једначини 10 за K_k на страни 5, држи у опсегу $[0, 3\pi/4)$ за дати косинус, за које време он константно опада. Односно, очигледно, $V(n)$ има неку врсту обрнуте сразмерности (не-линеарне, наравно) од K_k.

Наравно, овај проблем не важи за снагу, иначе Герцелов алгоритам не би имао смисла! Али, треба знати да је, у том смислу, Герцелов алгоритам лакше применити (треба нам мање бита) за више учестаности.

Као што рекосмо, Герцелов алгоритам рачуна један ДФТ резултат ("одбирак у фреквентном домену"). Али, ако га посматрамо као филтер, онда можемо да добијемо одговарајућу фреквентну карактеристику овог филтра. Слика ове карактеристике за Герцелов филтер на 2000Hz, у опсегу 1800 до 2200 Херца је дат на слици 2.2.

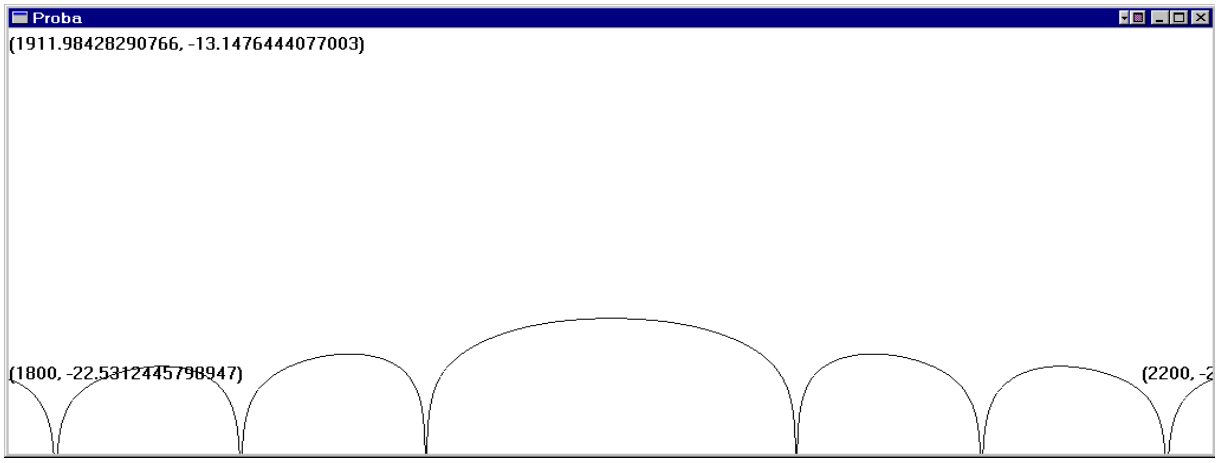

Figure 1: Фреквентна карактеристика Герцеловог филтра за 2000Hz у опсегу учестности 1800 до 2200 Херца

Слика је мало сиромашна, јер је скинута са екрана, из прозора програма који је нацртао, а који је Руби програм који управо служи да црта фреквентне карактеристике разних филтара. Наравно, има и специјализованих програма за то, али, овај је део развојног система који је направљен у ГВС-у. Рецимо, програм је интерактиван, па исписује вредности слабљења на произвољно изабраној учестаности (оној на којој се кликне мишем). Лепше слике могу да се нађу у књигама које описују ДФТ.

Елем, требало би да је ипак довољно јасно да се ради о скупу "трбуха", при чему је исти највиши (и најшири) око Герцелове учестаности, а ови остали се лагано смањују и ужи су и симетрични у односу на ту учестаност.

Међутим, овде може да се примети један проблем. Слабљење првих трбуха (лево и десно) је -12dB, што је премало за већину примена у препознавању тонова. Наиме, обично

се тражи већи распон нивоа које треба хватати. Без неких додатних радњи, Герцелов филтар за 2000Hz, са слике 2.2, би хватао и учестаности на 1912Hz, јер је потребан распон преко 20dB, а на слици (горњи леви угао) видимо да је слабљење на тој учестаности мање од -20dB (износи -13,14dB).

Постоје два могућа приступа. Један је да се одбирци провуку кроз "прозорску функцију". Прозорске функције, практично, представљају неку врсту филтера, али прављених управо са намером да реше овакве проблеме у резултатима ДФТ-а. Постоји неколико "популарних", пре свега Хамингов и Блекменов. Све ове функције слабе ове "споредне" трбухе, а већина помало ослаби и "главни" трбух, а све такође и шире главни трбух (чиме дакле шире пропусни опсег), а сужавају споредне. Хамингов прозор подоста слаби трбухе суседне главном, а онда остале мало мање, али, најмање слабљење споредних трбуха је -43dB. Хамингов прозор приближно дупло шири пропусни опсег. Блекменов прозор слаби све споредне трбухе приближно равномерно, најмање слабљење је око око -60dB, а главни трбух се шири приближно трипут.

Мада постоје и прозорске функције које не слабе главни трбух, оне су мање популарне јер веома шире исти, па тиме и пропусни опсег. У том смислу, математика за препознавање тонова бива сложенија, због чега се прозорске функције ретко користе у примени за препознавање тонова. То не значи да их не треба користити, али, значи да у овом напису нису даље разматране, већ је примењена друга идеја. Зовемо је "паметно слабљење".

Паметно слабљење донекле подсећа на аутоматску регулацију појачања. Наиме, пре проласка кроз Герцелов филтер, извршимо прорачун нивоа долазног сигнала (у пракси, саберемо квадрате свих одбирака, што је је нескалирана мера квадрата ефективног напона долазног сигнала). Онда, зависности од добијеног нивоа и потребног распона, вршимо слабљење долазног сигнала за одређени ниво. То се најчешће ради померањем свих одбирака удесно за одређени број бита (што чини дељење са 2, 4, 8, 16...).

Ваља приметити да паметно слабљење смањује потребу за "глупим" слабљењем. Наиме, ако сигнал паметно слабимо за, рецимо, највише 18dB, то приближно значи три бита. Дакле, ако нам је потребно слабљење од само три бита да би остали у опсегу при прорачуну Герцела, онда нам никакво глупо слабљење више не треба. Чак и да нам треба укупно слабљење од пет бита, ово паметно слабљење значи да нам треба глупо слабљење од само два бита. Употреба прозорских функција у принципу не смањује потребу за слабљењем због прекорачења опсега, мада би даља анализа, која није извршена, могла да покаже ипак и неку корист прозорских функција и у том смислу.

2.3 Примена

За примену су занимљиве неке ствари које су последица захтева за пријемнике и које су последица окружења (процесор и друго) на коме се примена врши.

2.3.1 Рачунања

Приметили смо да, у општем случају, није доста 16 бита за рачунање Герцеловог алгоритма. Шта онда да радимо? Мислим, пре свега имајући у виду да углавном користимо 16-битне ДСП-ове? Очигледно морамо да се довијамо, не можемо просто да пустимо математику да одради своје.

Герцелов коефицијент и рачунања V(n) Иако је, ради краћег писања, згодно писати само герцелов коефицијент, а не и његове делове, због тога што Герцелов коефицијент може да

буде већи од 1, за обраду је пожељније користити облик:

$$V(n) = 2cos_k V(n-1) - V(n-2) + x(n) \tag{19}$$

где је: $cos_k = \cos(2\pi \frac{f}{f_s})$. Дакле, cos_k је сигурно у опсегу -1 до +1, па једино остаје двојка која се постиже померањем улево за један, или сабирањем са самим собом. На пример, ево како овај рачун може да се изведе на ДСП 56166:

```
  ; y0 ::= kos, x0 ::= v_n1, b ::= v_n2, a ::= t
  ;; petlja (x1 ::= odbiraka)
  do x1,L1                ; for (; odbiraka > 0; --odbiraka) {
  mpy y0,x0,a             ;   t = kos * v_n1;
  asl a                   ;   t <<= 1 // daje: t = 2*kos*v_n1
  add y1,a  x:(r3)+,y1    ;   t += odbirak; odbirak = *pbyOdbir++
                          ;   // t == 2*kos*v_n1 + odbirak
  sub b,a                 ;   t -= v_n2
                          ;   // t == 2*kos*v_n1 - v_n2+odbirak
  tfr x0,b  a,x0          ;   v_n2 = v_n1, v_n1 = t
L1                        ; } // end for
```

Наравно, ово је само део целог алгоритма, али његово језгро. Ваља приметити да се овај код не бави прекорачењем, већ се ослања на то да ДСП56166 памти ако икад дође до препорачења, па проверу у том смислу врши након петље. Наравно, остаје нам задатак да пробамо да прекорачења избегнемо.

Прво, зависно од учестаности које треба да хватамо, можемо да проверимо колике се вредности за $V(n)$ добијају. Имамо оно Руби програмче од малопре, уз малу дораду:

```
v_n = v_n_1 = 0
v_n_max = 0
0.upto{N} { |n|
    v_n, v_n_1 = Kk*v_n - v_n_1 + x[n], v_n

    v_n_max = abs(v_n) if abs(v_n) > v_n_max
}
```

И тиме смо (у `v_n_max`) добили максимум који $V(n)$ добија при рачуну. Може нешто слично да се направи и у неком другом језику или у програму за унакрсне табеле.

Како год дошли до `v_n_max`, ако га поделимо са максимумом А-закона, можемо да добијемо однос, а заокруживши га на први већи степен двојке, добијамо колико нам битова треба "преко" А-закона. Кад смо већ кренули:

```
MAX_A_ZAKON = 1.52
visxak = Math.log10(v_n_max/MAX_A_ZAKON) / Math.log10(2.0) + 1
```

Искористили смо (надам се познат) идентитет: $\frac{\log_a x}{\log_a b} = \log_b x$.

Ако је `bita_viska` веће од три, онда морамо да одбацујемо најниже корисне бите долазног сигнала (пристиглих одбирака). Међутим, $V(n)$ не зависи само од учестаности, већ, пре свега, од нивоа долазног сигнала. Подсетимо се да А закон, у ствари, даје само четири бита податка о вредности сигнала, остало су знак и ниво. У том смислу, ми ћемо тиме да изгубимо корисне податке само за врло слабе сигнале, толико слабе, да их не треба препознати ни за један нормалан пријемник.

Посебно, као што већ напоменусмо, овде нам помаже "паметно" слабљење.

Кад смо већ кренули, ако претходни рачун `bita_viska` прогласимо за функцију (која враћа `bita_viska`), онда можемо да урадимо овако нешто:

```
MAX_A_ZAKON = 1.52 # negde definisano

def odrediBrojBitaVisxka(x)
    v_n = v_n_1 = 0
    v_n_max = 0
    x.each { |x_n|
        v_n, v_n_1 = Kk*v_n - v_n_1 + x_n, v_n
        v_n_1 = t;

        v_n_max = abs(v_n) if abs(v_n) > v_n_max
    }
    Math.log10(v_n_max/MAX_A_ZAKON) / Math.log10(2.0)
end

def oslabi(x, k)
    x.map! { |xn| xn / k }
end

x = []
## !! Ovde treba nekako popuniti x

slablyenye = 1
while odrediBrojBitaVisxka(x) > 0
    slablyenye *= 2
    oslabi(x, slablyenye)
end
```

Када се изврши овај код, добићемо (у `slablyenye`) потребно ослабљивање сигнала. Оно је степен двојке зато што нам је то најзгодније слабљење (у ДСП-у само померамо за тај степен).

Ово треба одредити за највиши ниво сигнала који треба да препознајемо (узимајући у обзир и паметно слабљење). За више нас “баш брига”. Наиме, морамо (у сваком случају) да водимо рачуна о прекорачењу у току рада, јер је тешко бити потпуно сигуран да смо паметним и глупим слабљењем заиста решили све могућности за прекорачење опсега. Ако нам се прекорачење ипак деси у току рачуна, треба да прекинемо извршавање Герцеловог алгоритма и (евентуално) пријавимо грешку. У том смислу ће нам се прекорачење десити за сигнале више од тог највишег, али то није проблем - тон неће бити препознат.

Треба приметити да је један бит слабљења у ствари слабљење за 6dB. То је илустровано у следећој табели:

YYY	dBm
111	+3
110	-3
101	-9
100	-15
011	-21
010	-28
001	-34
000	-40

YYY представљају одговарајућа три бита експонента из одбирка кодираног по А-закону.

Рачунања квадрата амплитуде Ако смо се некако изборили са $V(n)$, остаје нам коначни резултат. Сличним рачуницама као малопре може да се закључи да је квадрат амплитуде и до неколико стотина пута већа од $V(n)$ (СИ јединице су друге, али то нам, нумерички, ништа не значи). Где да нађемо још бита?

Овде може да се прибегне следећем трику. Ако смо заиста све време остали у опсегу 16 бита, пошто то представља, практично, запис у непокретном зарезу, онда смо, у тој представи, остали у опсегу $[-1, 1)$. У том опсегу, било које множење остаје у истом опсегу. У изразу за снагу имамо два квадрирања, која остају у опсегу, као и једно множење које остаје у опсегу до на множење са два. Шта се намеће? Па, да поделимо добијене $V(n)$ и $V(n-1)$ са два, па онда одрадимо математику која ће сигурно остати у опсегу $[-1, 1)$.

То неће бити тражени квадрат амплитуде, већ ослабљен, али, с друге стране, кога то занима, и тако смо у неком фиксном зарезу и морамо некако да "изађемо", па зар је важно са којим фактором? Треба приметити и да је то 16-битни процесор, па више од 16 бита не можемо да добијемо и да хоћемо. Тј., могли би да мувамо и ручно (дакле, са више наредби) да срачунамо снагу на 32 бита, па онда кренемо у поређење са тим, али, и та поређења би онда морала "ручно" да се праве. У сваком случају, 16 бита прецизности нам је довољно за већину потреба (они се пресликају у децибеле према приложеној табели).

Да то и математички представимо. Пођимо од једначине за снагу (12) и поделимо је са четири:

$$\frac{V_{mG}^2}{4} = \frac{V(n)^2}{4} + \frac{V(n-1)^2}{4} - \frac{K_k V(n) V(n-1)}{4}$$

Преуредимо:

$$\frac{V_{mG}^2}{4} = (\frac{V(n)}{2})^2 + (\frac{V(n-1)}{2})^2 - K_k \frac{V(n)}{2} \frac{V(n-1)}{2}$$

Укратко, ако поделимо добијене $V(n)$ и $V(n-1)$ са два, па онда срачунамо снагу по једначини 12, добићемо снагу подељену са четири. Дакле, уз све друге факторе које имамо због тога што радимо над 12-битним одбирцима на 16-битном процесору, код кога је 16 бита опсег $[-1, 1)$, треба да допишемо и фактор четири. Односно, у смислу раније добијене формуле за вредност резултата Герцела за одређени ниво долазног сигнала, једначину 16 треба поделити са четири.

Такође, морамо да узмемо у обзир и глупо слабљење. Наиме, њиме смо поделили долазни сигнал са неким бројем. Пошто је: $V_{eff} \sim V_m$, онда је: $V_m \sim x^2$, па је дакле и резултат Герцела сразмеран квадрату долазног сигнала. Ако је сигнал подељен са неком вредношћу, онда и резултат треба поделити са том вредношћу, дакле, добијамо следећу

једначину:

$$V_{mGs}^2 = \frac{0,3N^2 \cdot 10^{\frac{P_{dBm}}{10}}}{K_s} \tag{20}$$

V_{mGs}^2 је оно што ће да се добије на излазу Герцела када се на улазу доведе сигнал ослабљен (x_s) за коефицијент слабљења K_s.

Дакле, у практичне сврхе, једначина 20 је она коју треба срачунати "унапред" и добијену вредност поредити са резултатом Герцеловог алгоритма (филтра).

Ваља приметити да ово нема везе са паметним слабљењем. Паметно слабљење нам, пре свега, служи за решавање проблема малог распона слабљења трбуха Герцеловог филтра. Оно ће, додуше, да смањи потребу за глупим слабљењем, али, не утиче на једначину 20, јер не слаби све сигнале, већ само неке.

За крај, треба приметити да у пракси није примећен случај да су $V(n)$ и $V(n-1)$ у опсегу -1 до +1, а да онда V_{mG}^2 буде ван тог опсега. Ипак, није доказано да је то и немогуће, па је, док се то евентуално не докаже, сигурније ипак извршити овај захват "дељења са четири".

2.3.2 Паметно слабљење

Већ смо га описали, али, имамо оперативно питање како се одређује који сигнали се слабе, а који не. Поменули смо да за те потребе рачунамо суму квадрата одбирака, што је мера квадрата ефективног напона. Да будемо прецизни:

$$V_{eff}^2 = \frac{1}{N}\sum_{i=0}^{N} x_i^2$$

Ако заменимо V_{eff}^2 и мало преуредимо, добијамо:

$$NR_{ref}P_{mW} \cdot 10^{\frac{P_{dBm}}{10}} = \sum_{i=0}^{N} x_i^2$$

узимајући у обзир "глупо" слабљење, чисто нумерички добијамо:

$$\frac{0,6N \cdot 10^{\frac{P_{dBm}}{10}}}{K_s^2} = \sum_{i=0}^{N} x_{si}^2 \tag{21}$$

Дакле, границе за паметно слабљење, у ("миливатним") децибелима, треба прерачунати у суму квадрата (потенцијално ослабљених) одбирака према једначини 21, па онда тако добијени резултат користити за поређење.

2.3.3 Услови препознавања

Најчешће се препознају тонови од једне или две учестаности. Кад су две учестаности, онда је то избор од више могућности - постоји неки скуп учестаности које могу да се појаве, а ми треба да утврдимо које две од њих су се појавиле. Услови су најразличитији, али неке ствари су заједничке.

За сваку од учестаности је могуће извући, из захтева, податак о томе када мора и када не сме да се препозна. Ово се односи и на опсеге учестаности (нпр., $\pm 15Hz$), али и на опсеге нивоа (рецимо, -4 до -32 децибела).

За Герцелов алгоритам је најважнији услов препознавања, из кога се добија пропусни опсег, из кога се добија "прва апроксимација" броја одбирака (једначина 18). Као што

смо приметили, овај број одбирака може да се мења (смањи, најчешће, да би се брже препознало), али не значајно. Ту се често примењује трик "преклапања" - дакле, да је број одбирака који се пропуштају кроз алгоритам толики колико треба да буде, али да се то пропуштање ради чешће него што је тај број одбирака - након једне обраде, не одбаце се сви одбирци, већ се неки сачувају.

Нивои граница за препознавање се прерачунају по једначини 20 и онда користе у обради, али, прво је потребно одредити глупо слабљење. Глупо слабљење се одређује тако што се пусти сигнал Герцелове учестаности и максималног нивоа (+3dBm) и види колико треба да се ослаби тај сигнал.

Ту долазимо и до проблема, јер углавном постоји и захтев шта не сме да се препозна, што може да буде проблем да се задовољи са Герцеловим алгоритмом који је, као што смо приметили, филтер. Може да се деси да просто не постоји Герцелов филтер који може да испуни захтеве! Ту се примењује паметно слабљење и подешавање броја одбирака. По правилу, требало би да може да се дође до решења, ако захтеви нису превише строги.

Такође, и наш трик са слабљењем може да буде проблем, уколико постоји захтев да сигнал већег нивоа од задатог *не сме* да се препозна. Срећом, ово готово никад није захтев у препознавању тонова.

Посебне "погодности" настају за неке врсте двотона, код којих нису услови за све учестаности исти, а постоје и разне комбинације захтева, при чему, ради једноставности, по правилу узимамо исти број одбирака за све учестаности (наравно, морамо да пропустимо Герцелов алгоритам кроз све учестаности које се потенцијално јављају). Рецимо, ДТМФ пријемник је незгодан јер су учестаности све међусобно просте, па ниједан број одбирака не одговара свим учестаностима, већ треба наћи најмање лош, или правити сложену логику која синхронише више пријемника појединих тонова са различитим бројем одбирака. Ваља приметити да је у 21. веку постало широко познато да Герцелов алгоритам није пригодан за пуно испуњење захтева за ДТМФ пријемнике, управо због њихове строгоће. Ипак, и даље је Герцелов алгоритам најраспрострањенији и за ту примену, при чему се ваља помирити са тиме да неке захтеве просто неће испунити.

Све заједно, ово је класичан инжењерски проблем, у коме треба уз мало рачунања и пробања закључити шта је најмање лоше решење (које је можда и довољно добро). Понекад се може да дода још неки филтер, па онда да нешто и на основу њега закључимо. Наравно, да би ово могло да се ради у неком разумном времену, потребно је направити одговарајуће тестове (мисли се на одговарајуће датотеке са одбирцима) и онда их пропуштати кроз некако намештен алгоритам да би се видело да ли су захтеви задовољени или не.

2.3.4 Разлика нивоа два тона у двотону

Када се препознају двотони, један од параметара препознавања је да разлика нивоа два тона у двотону мора да буде мања од неке вредности, задате у децибелима. Наравно, пошто ми у Герцеловом алгоритму добијамо квадрат амплитуде, не можемо да просто одузимамо добијене вредности, већ нам опет треба нека математика. Ако ту разлику обележимо са P_{dBR}, онда имамо следеће извођење (под условом да је $V_{mG1} > V_{mG2}$):

$$P_{dBR} > 10 \log \frac{P_{G1}}{P_{G2}}$$

$$P_{dBR} > 10 \log \frac{V_{mG1}^2}{V_{mG2}^2}$$

$$10^{\frac{P_{dBR}}{10}} > \frac{V_{mG1}^2}{V_{mG2}^2}$$

Наравно, на ДСП-у нам по правилу не одговара дељење, па треба да преуредимо. Међутим, није проблем само у дељењу, проблем је што су вредности P_{dBR} увек веће од 1dB, па је лева страна једнакости увек већа од један, а ми радим са вредностима у опсегу -1 до +1. Дакле, треба се и са тим изборити. Ево даљег извођења:

$$\begin{aligned}
\frac{1}{10^{\frac{P_{dBR}}{10}}} &< \frac{V_{mG2}^2}{V_{mG1}^2} \\
10^{-\frac{P_{dBR}}{10}} &< \frac{V_{mG2}^2}{V_{mG1}^2} \\
10^{-\frac{P_{dBR}}{10}} \cdot V_{mG1}^2 &< V_{mG2}^2
\end{aligned}$$

Другим речима, треба одредити који је квадрат амплитуде већи, онда га помножити са $10^{-\frac{P_{dBR}}{10}}$ и то треба да буде мање (можда може и једнако, зависно од захтева) од другог квадрата амплитуде (резултата Герцела).