Annuaire de l'Universite de Sofia "St. Kliment Ohridski", Faculte de Physique, 100, 2007

АЛГОРИТЪМ ЗА ФАЗОВО ТЪРСЕНЕ

ЩЕРЬО ДЖИМОВ, СВЕТОСЛАВ ЦВЕТКОВ, ЮЛИЯН БУРОВ

Катедра "Физика на твърдото тяло и микроелектроника"

Щерьо Джимов, Светослав Цветков, Юлиян Буров. АЛГОРИТЪМ ЗА ФАЗОВО ТЪРСЕНЕ

В статията е описан и изследван алгоритъм за фазово търсене с оглед на приложението му за измерване на много малки периодични отмествания в звуковия диапазон при наличието на голям собствен или външен шум. Последният проблем е особено актуален при лазерните гравитационни обсерватории за детекция на гравитационни вълни. Предварително са разгледани понятията алгоритъм, фаза, амплитуда, честотен анализ, сигнали, осредняване на сигнали. Използвана е блокова структура за обяснение на самия алгоритъм, докато за анализ на неговата работа е използвана цифровата среда на Matlab.

Shteryo Dzhimov, Svetoslav Tsvetkov, Julian Burov. PHASE SEARCH ALGORITHM

An algorithm for phase search with a view to the application for measuring of very small periodic displacements in the sound frequencies at the presence of very high internal or external noise is described in the work. The last problem is especially important in the Laser Gravitational Observatories for the detection of Gravitational waves. The terms of algorithm, phase, amplitude, frequency analyses, signals and mean values are discussed in advance. A block structure for describing of the algorithm is used while for the analyses of its work a digital Matlab environment is utilised.

Keywords: laser interferometry, very small periodic displacements, gravitational waves, high noise and digital signal processing, computer simulation.

PACS number: 07.05.Tp

1. УВОД

В уводната част за удобство са разгледани основните величини от анализа на сигнали, използвани съществено в разработения от нас алгоритъм.

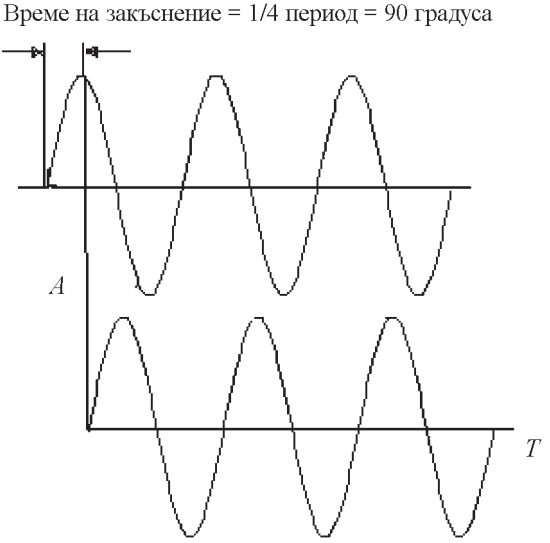
Алгоритъм

Алгоритъмът е процедура (пълен набор от добре дефинирани функции) за изпълнение на определена задача, която, зададена в едно начално положение, ще завърши в дефинирано крайно [1]. Алгоритмите се изразяват по различен начин, например с програмни езици. Ние използваме компютърна Matlab среда за писане и тестване на нашите алгоритми за обработка на сигнали. За да подобрим работата и бързодействието на алгоритмите ни, ние ги анализираме и изучаваме. Настоящият алгоритъм, който ще разгледаме, може да се класифицира като рекурсивен, сериен, детерминистичен, точен, евристичен, използващ намаляване и завоюване, търсещ, труден клас сложност (незавършен проблем във времето досега)

Концепция за фаза

Фазата е измерване на сравнително времево различие между две синусови вълни. Измерва се в ъгли с мярка градуси или радиани. Това представлява нормализирането на времето за един цикъл на вълната, без да се отнася до нейния истински времеви период [2].

Фазовото различие между две вълнови форми се нарича често *фазово отместване.* Фазово отместване на 2π rad е времево закъснение с 1 цикъл или един период на вълната, което не се счита за фазово отместване. Фазово отместване на $\pi/2$ rad е 1/4 от периода на вълната (фиг. 1) и т.н. Фазовото отместване може да е положително или отрицателно, т.е. една вълна може да изостава в сравнение с друга или да избързва. Тези условия са познати като фазово закъснение и фазово избързване.



Фиг. 1

Фазата може да бъде измерена към съответно време (с тригерен импулс).

Амплитуда на трептене

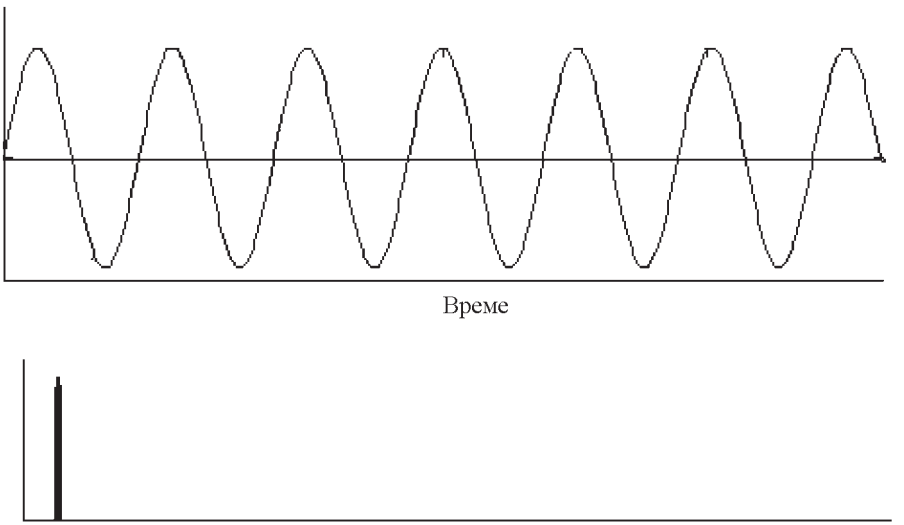
1. *Пикова амплитуда* (ПА) е максималното отклонение на вълната от нулева или равновесна точка. ПА до ПА (ПА-ПА) е разстоянието от отрицателен до положителен пик. В случая на синусова вълна стойността пик-до-пик е точно два пъти стойността на пика, защото вълновата форма е симетрична.

2. *Амплитуда корен от средното квадратично* (КСК, на английски RMS) е квадратен корен от квадратичните осреднени стойности на вълната. В случая на синусова вълна е 0.707 пъти от пиковата стойност. Стойността на тази амплитуда е пропорционална на площта под кривата – ако отрицателните пикове са изправени, т.е. станат положителни и областта под кривата се осредни до константа, нивото £ ще е пропорционално на тази амплитуда. КСК стойността на трептящ сигнал е важно измерване за неговата аплитуда. За да се определи, моментните амплитудни стойности на вълната трябва да се повдигнат на квадрат и да се осреднят за определено време. Този времеви интервал трябва да е поне един период на вълната, за да бъде вярна стойността. Всички квадратични стойности са положителни, както и техните осреднени. Тогава се извлича квадратният

корен на тази осреднена стойност, за да се получи КСК. КСК се използва във всички изчисления, касаещи сила и енергия на вълна.

Честотен анализ (спектрален анализ на трептящ сигнал)

$A(t)$ се нарича *вълна*, а $F(t)$ – *спектър*, като е в сила Time = 1 / Frequency; Frequency = 1 / Time. Отделните честотни компоненти са отделими и различими в спектъра и техните нива лесно се разпознават. Но е трудно да се извлече информация от времево разпределена трептяща вълна, защото тя има компоненти, невидими за окото. На фиг. 2–6 са показани примери за някои вълнови форми и техният спектър, които илюстрират важни характеристики на честотния анализ.

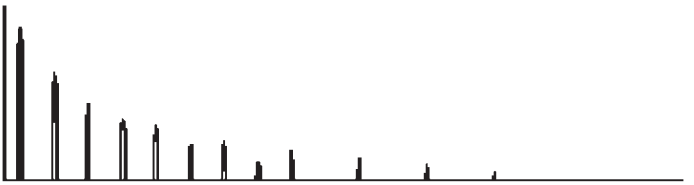


Спектър на синусова вълна

Фиг. 2

На фиг. 2 синусовата вълна се състои от една честота, а нейният спектър – от една точка. Математическата операция, която превръща вълна, разположена във времето, в такава в честотната област, е *фурие-трансформацията*.

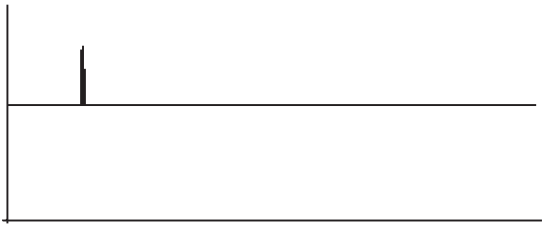
Време



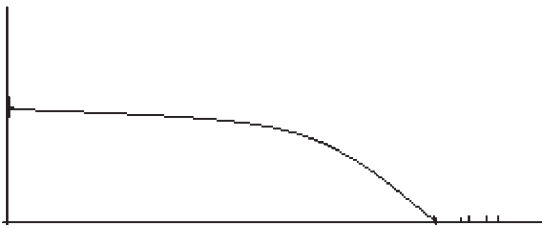
Честота

Фиг. 3

Всяка периодична вълна, която е симетрична, ще има спектър само с нечетни хармонични като на фиг. 3.



Време

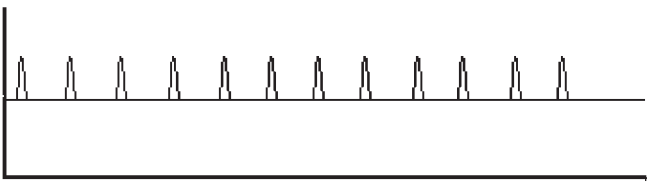


Честота

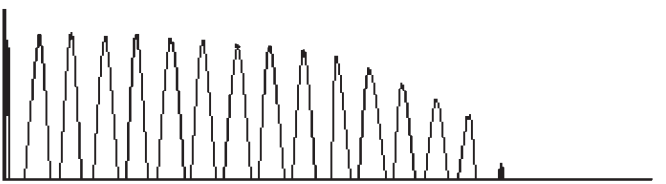
Фиг. 4

На фиг. 4 е показан къс импулс от сигнален генератор. Тук енергията на спектъра е разпределена продължително в голяма област от чес-

тоти, вместо да е концентрирана само в няколко. С това са характерни недетерминистични сигнали като случайни и кратки шумове. Характерно ограничение на фурие-анализа е, че не може да каже при разглеждане на продължителен спектър дали той не се дължи на случайни или преходни сигнали. В този случай се гледа и вълновата форма.



Време

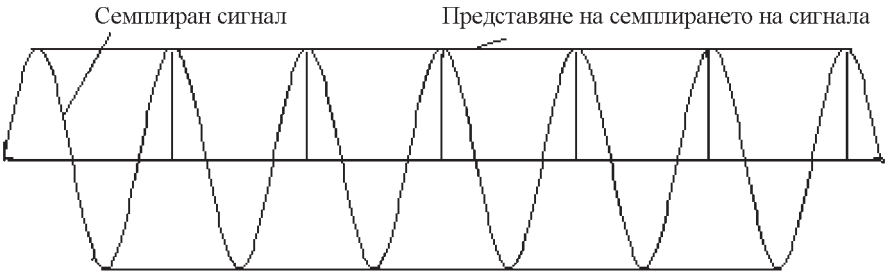


Честота

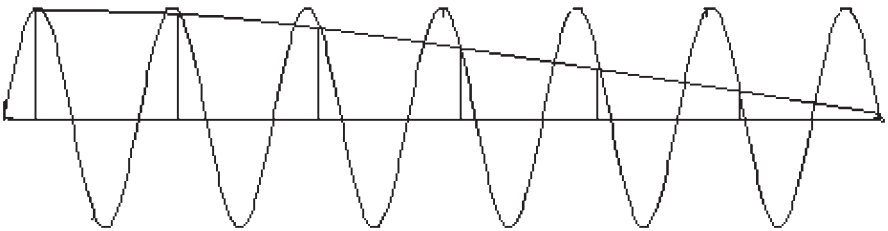
Фиг. 5

Повтаряне на един и същ импулс през определено време и неговият спектър са представени на фиг. 5. Процесите на *пулсиране* и *амплитудна модулация* се различават с разминаването им по фаза.

Проблемът на обединяване на честоти поради използване на честота, по-ниска от семплиращата, (на английски *aliasing*) е много съществен при обработката на сигнали в условия на силен шум (фиг. 6.).



Семплиране с честота, равна на семплиращата



Семплиране с честота, по-ниска от семплиращата

Фиг. 6

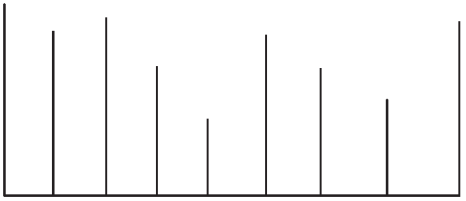
Типове сигнали

Стационарни – с еднакви статистически параметри във времето; амплитудното разпределение и стандартното отклонение са почти непроменяеми. Биват *случайни* и *детерминистични*. Случайните са непредсказуеми по честотно съдържание и ниво на амплитудата.

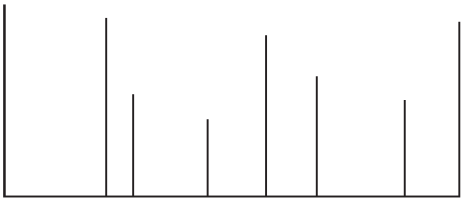
Детерминистични (фиг.7)

– *Периодичен* – компонентите са хармонични серии (хармониците са съставни на основната честота); образува спектър от дискретни честоти; шаблон от вълновата форма се повтаря на равни интервали от време.

– *Квазипериодичен* – няма хармонична връзка.



Периодични – компонентите са хармонични серии



Квазипериодични – без хармонична връзка

Фиг. 7

Нестационарни – продължителни или кратки във времето.

Осредняване на сигнала [2]

Говорим за *шум*, когато има нежелани флуктуации в измерваната стойност. Начинът за оптимизирано премахване на шум е чрез цифрова обработка на сигналите.

Кохерентно осредняване – процес на предетектиране, линейно или векторно осредняване, при което основна идея е разглеждане във времева скала, използвана за моделиране (семплиране) на оригиналния сигнал, т.е. събиране на много серии от сигнал с много шумови серии, като фазата на сигнала във времето във всяка серия трябва да е идентична. Например при осредняването на синусова вълна в шум кохерентното осредняване изисква фазата на вълната да е една и съща в началото на всеки измерен набор от семпъли. Когато е налице това улсовие, синусовата вълна ще бъде осреднена на нейната истинска синусова амплитудна стойност. Шумът е различен във всяка серия от семпъли и неговата средна стойност ще клони към нула. Идеята е, че кохерентното осредняване намалява шума и увеличава амплитудите на сигналите, които са синхронни или кохерентни с началото на семплиращия интервал.

Некохерентно осредняване – осигурява увеличена точност в измерването на относителни мощности на сигнала, което се използва в много

инструменти за тестване като спектрални, мрежови и сигнални анализатори.

Ние ще използваме *кохерентно осредняване* за детектиране на полезния сигнал, като това същевременно е и нашият критерий за полезен сигнал. Тъй като събирането е по-лесна и бърза операция от умножението, ние можем да осредним изходните стойности от множество бързи фуриерови трансформации (БФТ), за да осигурим по-добра чуствителност на БФТ. Увеличената БФТ-чуствителност и намаляването на шума се нарича още *печеливша интеграция*.

2. АЛГОРИТЪМ ЗА ФАЗОВО ТЪРСЕНЕ

В работи [4–6] сигналът от външен генератор е използван както за да извлече чрез *Lock in*-технологията подобен на него от общата смес на шум и слаб сигнал, така и за да разтрепти едното от огледалата на Фабри–Перо-интерферометър, което се използва и в технологиите за детектиране на гравитационни сигнали, притежаващи съвсем други параметри.

Нека разгледаме сигнали, които са резултат от непрекъснато протичащ за определено време физичен процес. Характерно за такъв тип сигнали е фуриеровите им компоненти да са с постоянна или бавнопроменяща се във времето фаза. В случай на подобен сигнал, потопен в шум, можем да запишем

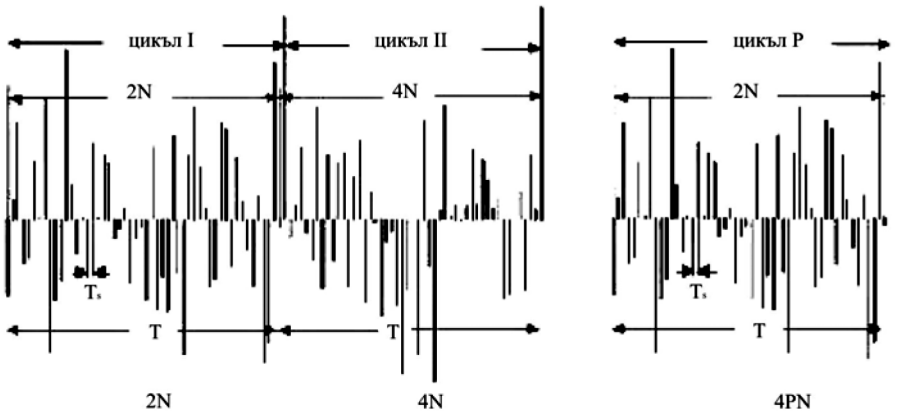
$$X(t) = \sum_{i=1}^{q} A_i \sin(\omega_i t + \varphi) + B \qquad (1)$$

където с A_i, $\omega_i = 2\pi f_i$ и φ_i са означени съответно i-тата амплитуда, кръговата честота и фазата на q на брой фуриерови компоненти на разглеждания сигнал, а с B – шумовата компонента в общия сигнал. Формулата е написана с фуриерови компоненти, за да представи по-общия математически запис, включващ в себе си възможните сложни сигнали като единични импулси, сигнали от вида на локомотивния, чуруликащия и др.

За да се приложи цифрова обработка на общия сигнал, последният се оцифрява, като се накъсва във времето с кръгова честота $\omega_s = 2\pi f_s = 2\pi / T_s$ така, че в точките $t = n/f_s = nT_s$ той получава стойностите $X(nT_s) = X_n = \sum_{i=1}^{q} A_i \sin\left(2\pi n \frac{T_s}{T_i} + \varphi_i\right) + B$, където n = 1, 2, 3,… е цяло положително число. Единственото условие при оцифряването, т.е. при прехо-

да от непрекъснат във времето към цифров сигнал (поредица от числа) и обратно, е да бъде спазена теоремата на Нейквист, т.е. неравенството $T_s < 12f_\alpha$, където $2\pi f_\alpha = \omega_\alpha$ е най-високата кръгова честота от съвкупността ω_i на кръговите фуриерови честоти на сигнала.

Като пример на фиг. 8 е показан цифров сигнал, съставен от синусов сигнал с честота $f = 1/T$, честота на дискретизация $f_s = 1/T_s$ и „бял“ гаусов шум.



Фиг. 8

И двата сигнала – гаусовият шум и потопеният в него синусов сигнал, са с еднакви амплитуди и с обща продължителност PT. Цялото положително число P се определя от броя на периодите T, които се нанасят в разглежданата обща дължина на сигнала, а n = 1, 2, 3,..$2N$. Всяка n-та стойност $X_n(nT_s)$ на общия сигнал е изобразена като вертикална отсечка (фиг. 8) съгласно формулата

$$X_m^{\text{sum}} = X_m \frac{T_s}{T} m + X_m \left(\frac{T_s}{T} m + 2N \right) + X_m \left(\frac{T_s}{T} m + 4N \right) + X_m \left(\frac{T_s}{T} m + 6N \right) +$$

$$+ \ldots + X_m \left(\frac{T_s}{T} m + P(2N) \right) = A \sum_{k=1}^{p} \sin \left(2\pi \left(\frac{f}{f_s} m + k \right) \right) + A \sum_{k=1}^{p} \text{randn}\left(\text{size}\left(n + k \right) T_s \right) =$$

$$= AP \sin \left(2\pi \frac{f}{f_s} m \right) + A \sum_{k=1}^{p} \text{randn}\left(\text{size}\left(n + k \right) T_s \right).$$

Функциите randn и size определят шумовата компонента, като randn генерира случайни числа с нормално разпределение в диапазон, определен от size. Функцията size връща размерността на даден вектор, ако като

параметър и бъде зададен скалар, size връща размерност 1×1. Както се вижда от първия член на получената, формула при кохерентното сумиране на компонентите на сигнала с една и съща фаза амплитудата на синусовия сигнал нараства P пъти, докато шумовата компонента поради некохерентното сумиране се очаква да нараства много по-бавно. Условието за кохерентно сумиране на хармоничен сигнал с честота f при честотно сканиране с честота F, изисква

$$\sin\left(2\pi\left(\frac{f}{f_s}m+(k+1)\frac{f}{F}\right)\right)=\sin\left(2\pi\left(\frac{f}{f_s}m+k\frac{f}{F}\right)\right).$$

Горното условие се реализира за сканираща честота F, свързана с f съгласно равенството:

$$\frac{f}{F}=s,$$

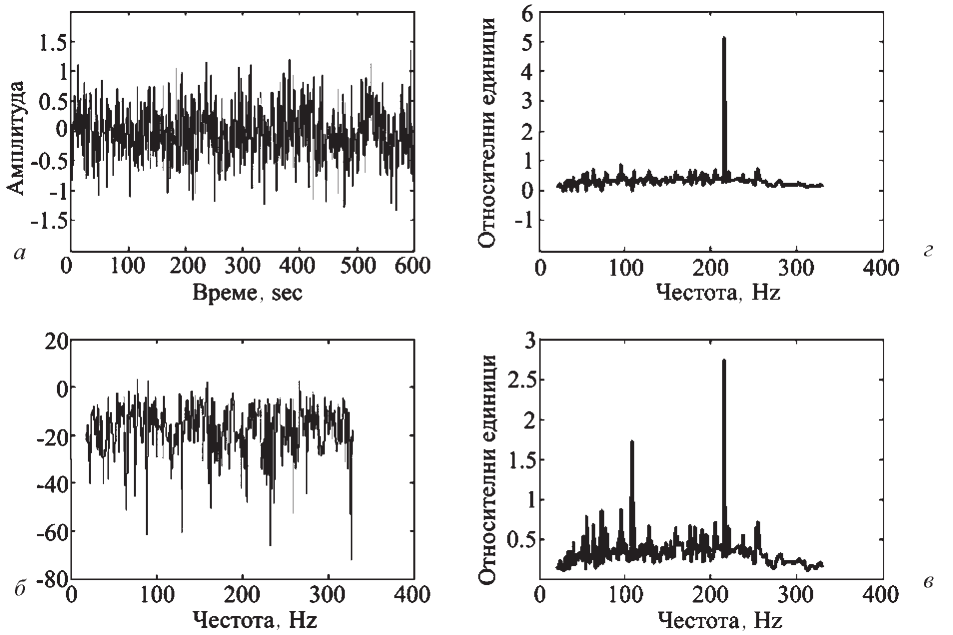
където s =1, 2, 3,… е цяло положително число.

Това са основните елементи, на разработения алгоритъм за определяне на много слаби сигнали на фона на силен шум.

По-долу е представен конкретен пример за работата на алгоритъма върху сигнал в силна шумова среда. Симулацията е извършена в MatLab среда. Генерираме сигнал с командния ред

$$X = 0.005*\sin(2*\text{pi}*216*n/21600)+0.5*\text{randn}(\text{size}(n)).$$

В него първият член представлява синусово трептене с честота f = 216 Hz и амплитуда A = 0.005 единици, смесен с бял гаусов шум (вторият член) с амплитуда 0.5 единици (девиация σ = 0.5) и средна стойност нула. Честотата на дискретизация е f_s = 21600 Hz. На фиг. 9*а–г* са показани съответно видът на самия сигнал, фуриеровото му честотно съдържание, етап от междинната обработка и крайният резултат. Виждат се (фиг. 9*в*) трептения с честоти 108 Hz, 72 Hz, 54 Hz, 36 Hz и т.н. На фиг. 9*г* е показан (след крайната обработка) ясно изразен сигнал на 216 Hz, превишаващ повече от 6 пъти околните сигнали.



Фиг. 9

В табл. 1 с удебелен шрифт са дадени командите, с които се дефинират променливи, извикват се вградени функции и др. Всяка от командите е резгледана подробно в раздела Help на Matlab програмата. Използвани са следните специални символи:

% – следва работен коментар,

; – край на команда.

Таблица 1

1. **B=3000;** %Брой на сумиранията, които се извършват за всяка една честота
2. **F=21600;** %честотата на семплиране, т.е. брой на семпалите за 1 секунда
3. **g=[0.93*22000/2,22000/2];** %от 3 до 8 ред включително са редове, засягащи %специално конструиран нискочестотен филтър, който се използва
4. **a=[1 0];** %при процедурата Resample от 25 ред. От тези редове се взимат за %Resample само коефициентите b.
5. **dev=[0.01 0.01];**
6. **freqz(b,1,1024,F);**
7. **[n,fo,ao,w]=remezord(g,a,dev,22000);**
8. **b=gremez(n,fo,ao,w);**
9. **x=wavread('waveF216HzN');** %Чете данни записани в wave %формат
10. **L = length(x);** %Числото L определя броя на елементите в редицата x

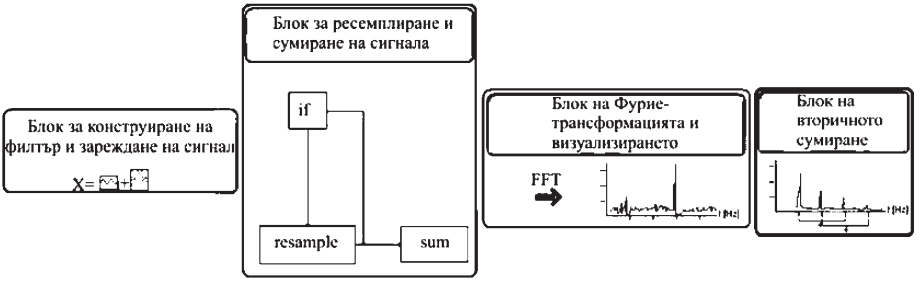
11. **for f=20:1:330;** %Чрез for се реализира цикъл, в които последователно се %извършват всички операции м/у for и съответстващото му end за всяко f от %20,21,22,...до 330
12. **Ef=F./f';** %Броя на семплите за един период, преди ресемплиране
13. **Z=round(Ef);** %Броя на семплите за един период, след ресемплиране
14. **Kf=Z./Ef;**
15. **if Kf == 1;**%Това са логически оператори. При изпълнение на това логическо %равенство се извършват всички действия след него, като се прескачат %операциите след elseif и се отива веднага на тези след първото end
16. **x=x(1:Ef*B);**%Оператор, с който се ограничава дължината на реда до числото %Ef*B
17. **z=reshape(x,Ef,B);**%Оператор, с който се преобразува дългият ред от 1 до %Ef*B, в матрица z с елементи EfxB
18. **v=rot90(z);**%Оператор, с който се завърта матрицата z на 90 градуса, така че %стълбовете стават редове и обратно. Това е нужно за действието на %следващия оператор
19. **r=sum(v)/B;** %Оператор, който сумира всички елементи от даден стълб
20. **elseif Kf ~= 1;**%Това са логически оператори. Ако не е изпълнено if, %изчисленията се прехвърлят тук на операторите след elseif. Тук са %всичките честоти, които не се делят целочислено на F. Затова тук се %извършва(продължава)
21. **x=x(1:Z*B);**%resample за всички честоти, които не се делят целочислено на F. %Тук се включва нискочестният филтър за изглаждане чрез коефициентите b, от %ред 4
22. **y=reshape(x,Z,B);** %Размерът на матрицата е {Z x B)}
23. **Frsf=Z.*f';** %Брой на семплите за 1 сек. След ресемплирането
24. **[p,q]=rat(Frsf./F,0.00001);** %rat е оператор, който представя цялото число %Frsf./F като частно на две най-малки цели числа с точност 0.00001. Това е %необходимо за операцията resample
25. **z = resample(y,p,q,b);** % Новите размери на матрицата са {Frsf x (L/B)}
26. **v=rot90(z);** %Отново завъртаме матрицата z, за да извършим сумиране на %елементите от нейните колони
27. **r=sum(v)/B;** % Новите размери на матрицата са (Frsf x 1)
28. **end**
29. **h=f/2;**
30. **s=round(h);**
31. **u=fft(r,s);** % Fast Fourier Transform.
32. **P=2*u.*conj(u)/f;** % Разпределение на мощностите.
33. **C(f)=max(P);** % максималните стойности.
34. **end** %Това end е в комбинация с for. След операциите if или elseif е %извършено пълно изчисление за амплитудата на една честота. Започва да се %обработва следващата, като се повтарят процедурите отново за новата f
35. **f=20:1:330;** %35 и 36 редове плотват изчислените дотук стойности на %мощността (амплитудата) за всички честоти от 20, 21, 22,...до 330 Hz
36. **plot(f,C(f));**

%Изчисленията, които се извършват до плотването, отнемат на компютъра %около 18 минути, а има нужда от допълнителна обработка на тези резултати, %тъй като алгоритъмът ни допуска да имаме максимуми и за всички други %честоти, за които е изпълнено условието fmax/f=1,2,3,4,...q, където с %fmax=216Hz и с f са означени съответно честота на полезния сигнал в %комплексния с голям шум сигнал. За това вместо операцията плотване аз %записах данните отново в wave-формат C(f) като функция на f. За това има

%втора част от обработка, която започва с четене на така записаното вече %разпределение а C(f) и по-нататъшна обработка на спектъра, за да изчистя %фундаменталното присъствие в него на fmax/f=1,2,3,4,...q, което изкривява %истинската спектрална зависимост.
%Ако тази първа част бъде пусната да се изпълнява от Mathlab, ще се види %как плотнатият спектър съдържа действително максимуми освен на 216 Hz още %и на 108 Hz, 72 Hz, 54 Hz и т.н., както това е показано на фиг. 9в
37. **C=wavread('MaxF216HzNN');** %Чете разпределението на C(f) от файл.
38. **f=20:1:330;**
39. **[D,d] = sortrows(C(f));**%Тук се извършва сортиране на стойностите на C(f) по %нарастващи стойности, т.е. най-голямите стойности са на %дъното на %колоната.
40. **f=flipud(d+19);%**Тази операция завърта реда от индексите на 180 градуса и %прибавя 19. Това се налага, защото индексите d от реда 47 започват от 0, %1, 2,... и завършват на 311 и за да се трансформират отново в мащаба на f
41. **e=flipud(sort(f(1:4)));** Отново сортирам този път f от 1 до 4 и ги завъртам %на 180 градуса. Те са подредени така 216 Hz, 108 Hz, 72H z и 54 Hz.
43. **if o==e(1)./e(1:4);**%Извършва се проверка за индексите от ред 49
42. **o=e(1)./e(1:4);**%операцията от 50 ред дели последователно индексите 216, %108, 72 и 54 един на друг и се проверява дали удоволетворяват условието %fmax/f=1,2,3,4,...Ако това е изпълнено следва, изпълнение на следващите %редове
44. **C(e(1))=sum(C(e(:)));**%Към C(216) се прибавят стойностите на C(108), C(72) и C(54)
45. **e=e(2:4);**%За индексите 2, 3 и 4 се извършва операцията от реда 54
46. **C(e)=mean(C(e-1)-C(e+1));**%Определят се нови стойности на C за индексите %108, 72 и 54 посредством средните стойности на C за съседните индекси
47. **f=20:1:330;** %Извършва се финалното плотване
48. **plot(f',C(f));** %.....
49. **y=wavread('daVinci MSpeakOK')**;%Генерира се звуков сигнал за успешно %намерен сигнал с постоянна фаза измежду големия шум на общия комплексен %сигнал
50. **wavplay(y,22000,'async');**
51. **elseif o~=e(1)./e(1:4);**%ако няма потвърждение на ред 51, това означава, че %не е намерен сигнал с постоянна фаза измежду големия шум на общия %комплексен сигнал%
52. **y=wavread('daVinci Error');**
53. **wavplay(y,22000,'async');**%Следва звуково съобщение на разочарование при %неуспех
54. **end**

3. СИМУЛАЦИЯ НА АЛГОРИТЪМА ЗА ФАЗОВО ТЪРСЕНЕ

Тестовете на алгоритъма за фазово търсене са извършени основно в MatLab среда, в която са написани симулационни програми, използващи арсенала от функции на MatLab библиотеките. След поредица от тестове, оптимизации и пренаписвания на симулационната програма се стигна до нейния краен-задоволителен вариант, в който програмният код има следната блокова структура:



В този блок се извършва конструирането на високочестотен филтър, чиито параметри ще бъдат използвани за ресемплиращата функция в блок 2 и зареждането от wav-файл на сигнал, потопен в дълбок шум с постоянна или бавно променяща се фаза.

Конкретен пример за осъществяването на този блок представлява следният код:

1. **B=3000;**
2. **F=21600;**
3. **g=[0.93*22000/2,22000/2];**
4. **a=[1 0];**
5. **dev=[0.01 0.01];**
6. **freqz(b,1,1024,F);**
7. **[n,fo,ao,w]=remezord(g,a,dev,22000);**
8. **b=gremez(n,fo,ao,w);**
9. **x=wavread('waveF216HzN');**

Ред 1. Инициализира променлива *B*, която ще бъде използвана в блок 2 .Тя съответства на брой на сумиранията, които се извършват за всяка една честота.

Ред 2. Инициализира променливата *F*, която съответства на семплиращата честота, с която е записан сигналът в wav-формат.

Ред 3–8. Извършват конструирането на високочестотен филтър, чиито параметри ще бъдат използвани в блок 2 от функцията за ресемплиране.

Ред 9. Зарежда сигнала от файл с име „waveF216HzN.wav“ във вектора ***X***.

Тук се извършва обхождане в цикъл на даден диапазон от честоти (в примера от 20 до 330 Hz). В цикъла чрез проверки са отделени два случая, когато дадената честота се дели на цяло число и когато не се дели на цяло число. И в двата случая векторът на сигнала бива трансформиран в матрица, чиито стълбове представляват отделните семпъли на сигнала. Следва сумиране по стълбовете на дадената матрица. Във втория случай се налага да бъде направен ресемпъл на матрицата на сигнала, защото семплите не са с дължина цяло число и това ще внесе грешка в изчисленията – няма да бъдат сумирани семпълите на една и съща честота.

Блокът се реализира по следния начин.

10. **for f=20:1:330**;
11. **Ef=F./f';**
12. **Z=round(Ef)**
13. **Kf=Z./Ef;**
14. **if Kf == 1;**
15. **x=x(1:Ef*B);** Оператор, с който се ограничава дължината на реда до числото Ef*B
16. **z=reshape(x,Ef,B);** Оператор, с който се преобразува дългият ред от 1 до Ef*B в матрица z с елементи EfxB
17. **v=rot90(z);** оператор, с който се завърта матрицата z на 90 градуса така че стълбовете стават редове
18. **r=sum(v)/B;** оператор, който сумира матрицата по стълбове
19. **elseif Kf ~= 1;**
20. **x=x(1:Z*B);**
21. **y=reshape(x,Z,B);**
22. **Frsf=Z.*f';**
23. **[p,q]=rat(Frsf./F,0.00001);** rat е оператор, който представя цялото число Frsf./F като частно на две най-малки цели числа с точност 0.00001
24. **z = resample(y,p,q,b);** ресемплираща функция
25. **v=rot90(z);** Отново завъртаме матрицата z, за да извършим сумиране на елементите от нейните стълбове
26. **r=sum(v)/B;**
27. **end**

Ред 10. Начало на цикъл, който обхожда честотите от 20 до 300 Hz.

Ред 11–13. Инициализират променливата *Kf*, която съдържа в себе си критерий за това дали текущата честота се дели на цяло число или не.

Ред 14. Извършва проверка на променливата *Kf* . Ако *Kf* = 1, тогава имаме случай на честота, деляща се на цяло число и програмата ще извърши сумиране по семпъли на сигнала, без да извършва ресемплиране.Тоест ще изпълни редовете от 15 до 18. Ако $Kf \neq 1$, тогава програмата изпълнява редовете от 20 до 26, т.е. текущата честота не се дели на цяло число и се налага след преобразуването на вектора на сигнала в матрица да бъде извършено ресемплиране (редове от 22 до 24). Следва сумиране по стълбове.

Ред 27. Край на проверките за *Kf*.

3.3. БЛОК НА ФУРИЕ -ТРАНСФОРМАЦИЯТА И ВИЗУАЛИЗИРАНЕТО

Резултатът от сумирането по семпъли бива запазен във вектор, на когото трябва да бъде направена фурие-трансформация. Всеки от фурие-коефициентите се умножава с неговото комплексноспрегнато и така за всяка една итерация от цикъла, се получава мощността на честотния спектър на сигнала. От всеки честотен набор се запазват във вектор максималните стойности. Стойностите на истинския сигнал след многократното сумиране на семпълите се увеличават много в сравнение с тези на шумовата компонента, което се дължи на бавно променящата се фаза на сигнала. Следва изобразяване на вектора с максималните стойности на честотния спектър. По оста *x* се разполагат честотите от избрания диапазон, а по оста *y* амплитудите на честотите в относителни единици. Поради посочените горе причини на графиката се очаква честотите, съдържащи се в истинския сигнал, да са с порядъци по-големи от тези на шумовата компонента.

По-долу е показан начин за осъществяването на блока.

28. **h=f/2;**

29. **s=round(h);**

30. **u=fft(r,s);** извършва фурие-трансформация върху вектор ***r*** и запазва коефициентите във вектор ***u***

31. **P=2*u.*conj(u)/f;** умножава всеки коефициент по комплексноспрегнатото му

32. **C(f)=max(P);** запазва максималната стойност от даден честотен набор във вектор ***C***

33. **end**
34**. f=20:1:330;**
35. **plot(f,C(f));** изобразява графика на честотното разпределение.

3.4. БЛОК НА ВТОРИЧНОТО СУМИРАНЕ

Обикновено след Фурие-трансформация се появяват максимуми и при честоти, които са кратни на честотите от истинския сигнал, т.е. отговарят на условието fmax/f = 1, 2, 3, 4,..., където fmax е честотата на полезния сигнал. На графиката, изчертана от ред 35 от кода на блок 3, могат да бъдат наблюдавани тези максимуми, кратни на главния максимум в графиката. Целта на блока на вторичното сумиране е да открие тези максимуми и да сумира техните стойности със стойността на истинския сигнал. По този начин амплитудата на истинският сигнал ще се увеличи още и така наречените кратни честоти няма да могат да бъдат тълкувани като честотни компоненти на полезния сигнал.

Пример за реализирането на блока

36. **f=20:1:330;**
37. **[D,d] = sortrows(C(f));**
38**. f=flipud(d+19);**
39. **e=flipud(sort(f(1:4)));**
40. **o=e(1)./e(1:4);**
41**. if o==e(1)./e(1:4);**
42. **C(e(1))=sum(C(e(:)));**
43. **f=20:1:330;**
44. **plot(f',C(f));**

Ред 41 извършва проверка дали индексите от реда 49 удоволетворяват условието fmax/f = 1, 2, 3, 4,... Ако това е изпълнено, следва изпълнение на следващите редове.

Редове от 36 до 39 извършват сортиране по големина на стойностите във вектора с максимумите ***C***, които след това се съхраняват във вектор d и съответстващите им честоти се подреждат във вектор ***D*.**

Редове от 40 до 41 извършват проверка дали индексите, запазени във вектора е (ред 39), удоволетворяват условието fmax / f = 1, 2, 3, 4,... Ако това е изпълнено, продължава изпълнението на следващите редове.

Ред 42 извършва вторичното сумиране и следва изчертаване на стойностите на вектора ***C*** от редовете f = 20:1:330 и plot(f',C(f))).

4. ЗАКЛЮЧЕНИЕ

– Създаден е и е тестван алгоритъм за фазово търсене в цифровата среда на Matlab, който работи коректно.
– Алгоритъмът има ограничение единствено от параметрите на хардуера, от който зависи времето му за изпълнение.
– Алгоритъмът допуска и е тествано разпределено мрежово изпълнение от няколко компютъра, за да се намали времето за изпълнение с оглед обработка в реално време на данните от лазерните гравитационни обсерватории или други източници.
– Алгоритъмът дава принос в изучаването на важни проблеми в цифровата комуникация и особено при цифровата обработка на резултатите за извличане на полезен сигнал в условията на силен външен или собствен шум.

Получените резултати дават възможност за:
– Използване и усъвършенстване на предложения алгоритъм за цифрово извличане на полезен сигнал в условията на силен външен или собствен шум.
– Свързване на доброволци и клъстери в мрежовата схема за разпределено изчисление.
– Използване на алгоритъма за създаване на софтуерени продукти за различни хардуерни платформи.

ЛИТЕРАТУРА

1. http://en.wikipedia.org/wiki/Algorithm
2. http://www.dliengineering.com/vibman/theconceptofphase.htm
3. Lyons, Richard G. Understanding digital signal processing / Richard G. Lyons. 2nd ed. p. cm. ISBN 0-13-108989-7; http://my.safaribooksonline.com/0131089897/ch11#X2ludGVybmFsX1RvYz94bWxpZD0wMTMxMDg5ODk3L2NvcHlyaWdodHBn
4. Borissov, M. I. and J. I . Burov, "One-arm Phase Optical Bridge with Axial Symmetry [with Interference Newton Frenges] for Measuring Small Periodic Displacements", *Electron. Letters*, 9, 355–357, 1973.
5. Borissov, M., Kl. Brunsalov and J. Burov, „One-arm Phase Optical Bridge for Measuring Small Amplitude High Frequency Vibrations" J*ap. J. Appl. Physics,* 15, 797–801, 1976.
6. Burov, J. and D. V.Ivanov, „Multibeam Interferometric Methods for Measuring Very Small Periodic Displacements", *Applied Optics,* 28 , 3343–3350, 1989.
7. Буров, Ю. „Детекция с лазер на много малки периодични премествания – новият път за изследване на вселената", *Университетско издателство „Епископ Константин Преславски"*. Шумен , 72–87, или http://elearning-phys.uni-sofia.bg/csus/index_files/inmem.pdf

8. J. Burov, S. Dzhimov, S. Tsvetkov," EXTRACTING THE USEFUL SIGNAL FROM HIGHLY NOISED BACKGROUND", 6th International Conference of Physics, Istanbul, *AIP publications*, 2007.
9. Miller, I., and Freund, J. *Probability and Statistics for Engineers*, 2nd Ed., *Prentice-Hall*, Englewood Cliffs. New Jersey, p. **118,** 1977.
10. Beller, J., and Pless, W. „A Modular All-Haul Optical Time-Domain Reflectometer for Characterizing Fiber Links," *Hewlett-Packard Journal*, February 1993.
11. Spiegel, M. R. *Theory and Problems of Statistics*, Shaum's Outline Series, McGraw-Hill Book Co. New York, 1961, p. 142.
12. Papoulis, A. *Probability, Random Variables, and Stochastic Processes*, *McGraw-Hill Book Co.* New York, 1984, p. 245.
13. Davenport, W. B., Jr., and Root, W. L. *Random Signals and Noise*, *McGraw-Hill Book Co*. New York, 1958, 81–84.
14. Welch, P. D. "The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified Periodograms", *IEEE Transactions on Audio and Electroacoust.*, Vol. AU-15, No. 2, June 1967.
15. Harris, F. J. „On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, Vol. 66, No. 1, January 1978.
16. Booster, D. H. , et al. „Design of a Precision Optical Low-Coherence Reflectometer," *Hewlett-Packard Journal*, February 1993.
17. Witte, R. A. „Averaging Techniques Reduce Test Noise, Improve Accuracy," *Microwaves & RF*, February 1988.
18. Oxaal, J. „Temporal Averaging Techniques Reduce Image Noise," *EDN*, March 17, 1983.
19. Lymer, A. „Digital-Modulation Scheme Processes RF Broadcast Signals," *Microwaves & RF*, April 1994.

Постъпила декември 2006 г.

Проф. Юлиян Буров
Софийски университет „Св. Климент Охридски"
Физически факултет
Катедра „Физика на твърдото тяло и микроелектроника"
Бул. „Джеймс Баучър" 5
1164 София, България
E-mail: burov@phys.uni-sofia.bg