Rekonstrukcja zaszumionego sygnału sinusoidalnego na podstawie niewielkiej liczby próbek za pomocą algorytmu ewolucyjnego

Piotr Kardasz

Wydział Elektryczny, Politechnika Białostocka

Streszczenie: Artykuł przedstawia przykład wykorzystania algorytmu ewolucyjnego w celu rekonstrukcji zakłóconego szumem białym sygnału sinusoidalnego na podstawie niewielkiej liczby losowo pobranych próbek tego sygnału. Wyniki eksperymentu pokazują, że proponowany algorytm ewolucyjny jest w stanie zidentyfikować z dużą dokładnością parametry badanego sygnału nawet przy dużej mocy zakłócającego szumu (SNR = 0 dB) Zbadane zostało również zachowanie się proponowanego algorytmu, jego zbieżność i dokładność otrzymanych wyników w zależności od parametrów rekonstruowanego sygnału. Zarysowane zostały również kierunki dalszych badań.

Słowa kluczowe: algorytm ewolucyjny, rekonstrukcja sygnału

1. Wprowadzenie

W ostatnich latach szybko rozwija się metoda próbkowania i kodowania sygnałów zwana Compressed Sensing. W odróżnieniu od klasycznych metod, polegających na pobraniu dużej liczby próbek, które są następnie poddawane obróbce, metoda ta pozwala na rekonstrukcję sygnału na podstawie niewielkiej liczby jego próbek.

Rekonstrukcja sygnału na podstawie niewielkiej liczby próbek może być użyteczna w sytuacjach, gdy pobranie wszystkich próbek niezbędnych do rekonstrukcji takiego sygnału zgodnie z twierdzeniem Kotielnikowa-Shannona o próbkowaniu [2] przekracza możliwości techniczne wykorzystywanego sprzętu [7, 8]. Rekonstrukcja taka może stać się przydatna również w warunkach, gdy z powodu zakłóceń nie dysponujemy wszystkimi wartościami próbek danego sygnału.

Proces rekonstrukcji sygnału polega na odnalezieniu „właściwego" rozwiązania spośród nieskończonej liczby rozwiązań niedostatecznie określonego układu równań i jest bardzo złożony obliczeniowo [1]. Jedną z możliwości rozwiązywania problemów o dużej złożoności obliczeniowej jest zastosowanie algorytmów inteligentnych, takich jak opisane w [3, 4] algorytmy genetyczne i ewolucyjne. W tej sytuacji celowym wydało się przetestowanie możliwości zastosowania takich algorytmów w celu rekonstrukcji tak spróbkowanych sygnałów.

Podczas procesu analogowego przetwarzania sygnału wprowadzane są do niego szumy i zakłócenia. Są to zarówno szumy zastosowanego czujnika, jak i szumy analogowych układów wzmacniających sygnał i dopasowujących go do standardów wybranego kanału transmisyjnego. Jeśli mamy do czynienia z analogowym kanałem transmisyjnym, on również wprowadza do sygnału szumy i zakłócenia. Zakłócony sygnał nie może być opisany niewielką liczbą współczynników, co jest podstawowym założeniem leżącym u podstawy metody Compressed Sensing [1]. Podejmowane są prace mające na celu rozwiązanie tego problemu [5, 6]. W tej sytuacji celowym wydało się również sprawdzenie, czy algorytm ewolucyjny jest w stanie wykryć i zrekonstruować sygnał zakłócony addytywnym szumem szerokopasmowym.

2. Cel i zakres przeprowadzonych badań

Celem przeprowadzonych badań było sprawdzenie, czy i z jaką dokładnością algorytm ewolucyjny jest w stanie zrekonstruować sygnał w postaci:

$$x(t) = A\sin(2\pi ft + \varphi) \tag{1}$$

do którego został dodany biały szum. Jako sygnał testowy użyty został sygnał sinusoidalny o częstotliwościach 50, 250 i 1250 Hz, wygenerowany z częstotliwością próbkowania 44100 Hz. Do sygnału dodany został szum biały o różnej mocy odpowiadającej określonym wartościom współczynnika stosunku sygnału do szumu (SNR = 20 dB, 3 dB, 0 dB). Pobrano 2048 kolejnych próbek, co odpowiada 21,53 milisekundom sygnału. Wylosowano spośród nich 8, 32 i 128 próbek. Algorytm ewolucyjny miał za zadanie zrekonstruować z tych próbek pierwotny, nie zaszumiony sygnał. Dla porównania rekonstrukcji został poddany również sygnał pozbawiony szumu oraz zbiór wszystkich 2048 próbek sygnału.

Wyniki eksperymentów zostały przedstawione w postaci wykresów i tabel.

3. Algorytm ewolucyjny

Program implementujący algorytm ewolucyjny opracowany w celu przetestowania wymienionych powyżej hipotez

został napisany w środowisku uruchomieniowym Lazarus. Przeprowadzone zostały próby jego działania, na podstawie których zostały wstępnie oszacowane podstawowe parametry algorytmu, takie jak wielkość populacji i liczba kroków obliczeń. Wstępnie dostrojony model został następnie zastosowany w celu przeprowadzenia właściwych badań.

Zastosowany algorytm ewolucyjny operował na obiektach („chromosomach") o trzech elementach, reprezentujących wielkości *A, f, φ*. Dodatkowo zarezerwowano w chromosomie zmienną przechowującą wartość jego funkcji przystosowania. Nie miało to wpływu na działanie samego algorytmu, ale pozwoliło na uproszczenie implementacji procedury, dokonującej selekcji.

Jako funkcja przystosowania został użyty średni moduł różnicy pomiędzy wartościami sygnału, wygenerowanego na podstawie zapisanych w chromosomie wartościach *A, f, φ* w chwilach, w których pobrano próbki, a wartościami tych próbek.

$$F = \frac{1}{N}\sum_{i=1}^{N}\left|x(t_i) - x_r(t_i)\right| \qquad (2)$$

gdzie $x(t_i)$ - wartość próbki oryginalnego sygnału, a $x_r(t_i)$ - wartość próbki sygnału zrekonstruowanego.

Wynik działania algorytmu jest tym lepszy, im mniejsza jest wartość funkcji (2). W szczególności, jeśli zostanie dokładnie zrekonstruowany sygnał pozbawiony szumów, funkcja przystosowania *F* przybierze wartość 0. W związku z tym zastosowany w eksperymencie algorytm ewolucyjny ma za zadanie dążyć do minimum funkcji przystosowania (2).

W drodze wstępnych eksperymentów wielkość podstawowej populacji została określona jako 1024 elementy. Dodatkowo zarezerwowane zostało również miejsce na 32 elementy – „rodziców", wybieranych na podstawie wartości ich funkcji przystosowania. Z elementów tych metodą krzyżowania „każdy z każdym" generowane było następne pokolenie. Łącznie populacja eksperymentalna zawierała więc 1056 elementów.

Przed rozpoczęciem pracy algorytmu ewolucyjnego program generuje 2048 próbek zadanej funkcji (1), dodając do nich (jeśli takie było założenie eksperymentu) wartości pobrane z generatora liczb pseudolosowych, pomnożone przez wybrany współczynnik. Pseudolosowe próbki symulują szum biały, dodany do sygnału. Następnie pobierana jest losowo z sygnału zadana przez eksperymentatora liczba próbek. Zapisywany jest wylosowany numer próbki z przedziału <0,2047> oraz jej wartość.

Algorytm ewolucyjny rozpoczyna pracę od wylosowania wartości A_k, f_k, φ_k dla każdego elementu *k* populacji. Następnie, dla każdego elementu populacji, generowanych jest 2048 próbek funkcji

$$x_k(t) = A_k \sin(2\pi f_k t + \varphi_k) \qquad (3)$$

W następnym kroku dla każdego elementu populacji jest obliczana wartość jego funkcji przystosowania (2). Na podstawie tej wartości przeprowadzana jest selekcja. Polega ona na odnalezieniu w populacji 32 obiektów o najniższych wartościach funkcji przystosowania. Obiekty te stają się „rodzicami" następnego pokolenia. Tworzone jest ono w ten sposób, że dla każdej pary elementów są obliczane wartości parametrów sinusoid, reprezentowanych przez nowe elementy populacji, na podstawie następujących wzorów:

$$A_{k+32\cdot l} = 0{,}5(A_k + A_l)\left(1 - \frac{m}{2} + m \cdot r\right) \qquad (4)$$

$$f_{k+32\cdot l} = 0{,}5(f_k + f_l)\left(1 - \frac{m}{2} + m \cdot r\right) \qquad (5)$$

$$\varphi_{k+32\cdot l} = 0{,}5(\varphi_k^* + \varphi_l^*) + 2\pi m r - \pi m \qquad (6)$$

gdzie *m* jest współczynnikiem mutacji, a *r* liczbą pseudolosową z zakresu <0,1)

Wartości amplitudy, częstotliwości i fazy potomków są więc średnią arytmetyczną tych wartości dla rodziców, zmienioną o pewną losową wielkość – „mutację".

W celu obliczenia nowej wartości fazy, ze względu na okresowość funkcji sinusoidalnej, celowa jest wstępna korekta faz rodziców φ_k, φ_l poprzez dodanie lub odjęcie od jednej z nich wartości kąta pełnego, tak, aby różnica między nimi nie przekraczała π. Uśrednieniu podlegają otrzymane po korekcie wartości φ_k^*, φ_l^*.

Wstępne testy pokazały, że badany algorytm ewolucyjny działa znacznie sprawniej, jeśli zostanie wprowadzony współczynnik mutacji malejący z każdym pokoleniem. Jeśli współczynnik ten jest duży, algorytm szybko odnajduje rozwiązanie bliskie zadanemu sygnałowi, jednak w kolejnych pokoleniach rezultat ten nie ulega poprawie. Jeśli zaś współczynnik ten jest mały, algorytm częściej wpada w „pułapkę" lokalnych minimów, a poza tym jego zbieżność jest niewielka, w zamian za to możemy jednak otrzymać bardziej precyzyjne rozwiązanie. Przeprowadzone próby zastosowania zmiennego w czasie współczynnika mutacji pokazały, że algorytm działa wtedy sprawniej, łącząc szybkie osiąganie punktu bliskiego rozwiązaniu z precyzyjnym „dopasowywaniem się" do niego.

Na skutek krzyżowania 32 elementów metodą „każdy z każdym" powstają 1024 nowe elementy. Pula ta jest uzupełniana o rodziców, co zapewnia, że w każdym pokoleniu funkcja przystosowania będzie zawsze mniejsza lub równa tej funkcji dla poprzedniego pokolenia. W ten sposób uzyskujemy nową pulę 1056 elementów stanowiących nowe pokolenie, dla którego procedura jest powtarzana.

Algorytm nie ma kryterium automatycznego stopu. Liczba obliczanych pokoleń została zadawana przed uruchomieniem obliczeń.

Program implementujący powyżej opisany algorytm ewolucyjny został napisany w języku Object Pascal w środowisku uruchomieniowym Lazarus. Środowisko to jest dostępne bezpłatnie na licencji LGPL.

4. Przebieg i wyniki badań

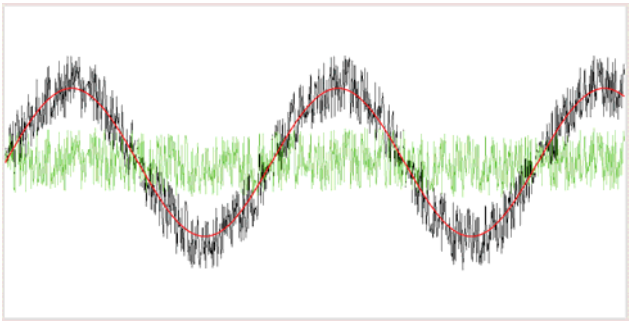
4.1. Sygnały testowe i ich próbkowanie

W celu przeprowadzenia badań zostały wybrane sygnały opisane w tabeli 1. Wstępne eksperymenty wykazały, że zmiana wartości amplitudy oraz fazy sygnału nie ma większego wpływu na uzyskane wyniki. Dla każdego sygnału została więc wybrana amplituda równa 1, oraz faza równa 0.

Dla każdego z sygnałów wylosowano zestawy 8, 32 i 256 próbek. Dla porównania został również użyty zbiór 2048 próbek (pełna reprezentacja sygnału). Rys. 1 pokazuje przebiegi, generowane przez program podczas pracy.

Tab. 1. Parametry sygnałów użytych w eksperymencie
Tab. 1. Experimental signal parameters

Lp	Częstotliwość f [Hz]	SNR [dB]
1	50	∞
2	50	20
3	50	3
4	50	0
5	250	∞
6	250	20
7	250	3
8	250	0
9	1250	∞
10	1250	20
11	1250	3
12	1250	0



Rys. 1. Przebiegi generowane podczas pracy algorytmu ewolucyjnego. Kolory: czarny – oryginalny sygnał, czerwony - zrekonstruowany sygnał, zielony – różnica między sygnałem oryginalnym i zrekonstruowanym
Fig. 1. Waveforms generated in the evolutionary algorithm. Colors: black - the original signal, red - the reconstructed signal, green - the difference between the original and the reconstructed signals

Kolorem czarnym narysowany jest badany sygnał, czerwonym sygnał wygenerowany przez najlepiej przystosowany element populacji, zielonym – różnica między tymi sygnałami.

4.2. Wyniki badań

4.2.1. Problem określenia minimalnej liczby próbek

Opisane w [1] metody pozwalają na odtworzenie sygnału o N składowych sinusoidalnych na podstawie co najmniej $2N$ jego losowo pobranych próbek. Testowany algorytm ewolucyjny nie był jednak w stanie zrekonstruować czystego sygnału sinusoidalnego na podstawie jego dwóch losowo pobranych próbek. Podczas wstępnych testów udało się zrekonstruować czysty sygnał sinusoidalny na podstawie jego pięciu próbek.

Problem określenia minimalnej liczby próbek sygnału, niezbędnej do jego rekonstrukcji przez testowany algorytm pozostaje tematem do dalszych badań.

4.2.2. Wyniki eksperymentów

Dla każdego zestawu częstotliwości i poziomu szumów uruchomiono opisany algorytm ewolucyjny, który pracował przez 256 pokoleń. Tabele 2, 3, 4 i 5 przedstawiają błędy rekonstrukcji sygnału odpowiednio dla $N=8$, 32, 256 i 2048 próbek, przy czym

$$\Delta A = 100\left|\left(\frac{A_r - A}{A}\right)\right| \tag{7}$$

$$\Delta f = 100\left|\left(\frac{f_r - f}{f}\right)\right| \tag{8}$$

$$\Delta \varphi = 100\left|\left(\frac{(\varphi_r - \varphi)\bmod 2\pi}{2\pi}\right)\right| \tag{9}$$

gdzie A, f, φ - parametry sygnału testowego, A_r, f_r, φ_r – parametry sygnału zrekonstruowanego.

Tab. 2. Błędy parametrów zrekonstruowanego sygnału dla N=8
Tab. 2. Reconstructed signal parameters error for N=8

Lp	f [Hz]	SNR [dB]	ΔA [%]	Δf [%]	$\Delta\varphi$ [%]
1	50	∞	$1{,}30\cdot10^{-3}$	$9{,}83\cdot10^{-5}$	$1{,}19\cdot10^{-4}$
2	50	20	2,78	0,827	1,60
3	250	∞	$4{,}85\cdot10^{-3}$	$2.24\cdot10^{-4}$	$1.91\cdot10^{-3}$
4	250	20	5,66	0,471	2,69
5	1250	∞	$3{,}73\cdot10^{-4}$	$6{,}25\cdot10^{-5}$	$2{,}67\cdot10^{-3}$
6	1250	20	8,56	$3{,}52\cdot10^{-2}$	1,70

Dla $N=8$ próbek algorytm nie był w stanie dokonać rekonstrukcji zaszumionego sygnału przy SNR = 3 dB i SNR = 0 dB

Tab. 3. Błędy parametrów zrekonstruowanego sygnału dla N= 32
Tab. 3. Reconstructed signal parameters error for N= 32

Lp	f [Hz]	SNR [dB]	ΔA [%]	Δf [%]	$\Delta\varphi$ [%]
1	50	∞	$1{,}28{\cdot}10^{-3}$	$6{,}35{\cdot}10^{-4}$	$1.00{\cdot}10^{-3}$
2	50	20	1,19	$5{,}00{\cdot}10^{-2}$	$1{,}56{\cdot}10^{-3}$
3	50	3	8,36	0,385	$6{,}84{\cdot}10^{-3}$
4	50	0	niepowodzenie (f_r=1181 Hz)		
5	250	∞	$4{,}32{\cdot}10^{-3}$	$4{,}62{\cdot}10^{-4}$	$2{,}77{\cdot}10^{-3}$
6	250	20	5,12	0,240	1,07
7	250	3	57,4	1,64	7,22
8	250	0	86,7	1,96	8,86
9	1250	∞	$6{,}30{\cdot}10^{-4}$	$9{,}18{\cdot}10^{-5}$	$4{,}61{\cdot}10^{-3}$
10	1250	20	0,478	$4{,}67{\cdot}10^{-2}$	1,29
11	1250	3	4,17	0,325	9,04
12	1250	0	9,40	0,461	13,1

Tab. 4. Błędy parametrów zrekonstruowanego sygnału dla N= 256
Tab. 4. Reconstructed signal parameters error for N= 256

Lp	f [Hz]	SNR [dB]	ΔA[%]	Δf[%]	$\Delta\varphi$[%]
1	50	∞	$9{,}58{\cdot}10^{-4}$	$7{,}10{\cdot}10^{-5}$	$9{,}69{\cdot}10^{-5}$
2	50	20	0,620	$3{,}59{\cdot}10^{-2}$	0,139
3	50	3	4,54	0,245	0,950
4	50	0	6,48	0,322	1,32
5	250	∞	$4{,}93{\cdot}10^{-4}$	$1{,}00{\cdot}10^{-4}$	$3{,}92{\cdot}10^{-4}$
6	250	20	1,25	$6{,}84{\cdot}10^{-2}$	0,533
7	250	3	8,90	0,465	3,66
8	250	0	12,7	0,648	5,13
9	1250	∞	$5{,}37{\cdot}10^{-5}$	$2{,}93{\cdot}10^{-5}$	$5{,}09{\cdot}10^{-4}$
10	1250	20	0,642	$9{,}90{\cdot}10^{-4}$	$6{,}52{\cdot}10^{-2}$
11	1250	3	3,54	$8{,}37{\cdot}10^{-3}$	0,485
12	1250	0	0,156	$2{,}55{\cdot}10^{-2}$	0,916

Tab. 5. Błędy parametrów zrekonstruowanego sygnału dla N=2048
Tab. 5. Reconstructed signal parameters error for N=2048

Lp	f [Hz]	SNR [dB]	ΔA[%]	Δf[%]	$\Delta\varphi$[%]
1	50	∞	$3{,}35{\cdot}10^{-4}$	$9{,}96{\cdot}10^{-5}$	$4{,}16{\cdot}10^{-5}$
2	50	20	0,366	$9{,}92{\cdot}10^{-3}$	$3{,}23{\cdot}10^{-2}$
3	50	3	2,58	$6{,}18{\cdot}10^{-2}$	0,218
4	50	0	3,66	$8{,}66{\cdot}10^{-2}$	0,305
5	250	∞	$1{,}47{\cdot}10^{-3}$	$3{,}27{\cdot}10^{-4}$	$1{,}92{\cdot}10^{-3}$
6	250	20	0,522	$1{,}92{\cdot}10^{-2}$	0,167
7	250	3	3,74	0,129	1,11
8	250	0	5,28	0,181	1,55
9	1250	∞	$2{,}19{\cdot}10^{-3}$	$4{,}83{\cdot}10^{-6}$	$2{,}56{\cdot}10^{-4}$
10	1250	20	$1{,}03{\cdot}10^{-2}$	$3{,}99{\cdot}10^{-3}$	0,169
11	1250	3	6,42	$2{,}38{\cdot}10^{-2}$	1,03
12	1250	0	0,312	$3{,}79{\cdot}10^{-2}$	1,64

W celu bardziej precyzyjnego ukazania zależności dokładności rekonstrukcji sygnału od parametrów takich jak liczba wylosowanych próbek i stosunek mocy sygnału do mocy szumu, badany algorytm został uruchomiony dla f= 250 Hz i SNR = 20 dB przy różnych liczbach próbek, oraz dla f= 250 Hz i N= 2048 dla różnych wartości SNR. Wyniki przedstawiają tabele 6 i 7.

Tab. 6. Błędy parametrów zrekonstruowanego sygnału dla f= 250 Hz i SNR= 20 dB dla różnych wartości N
Tab. 6. Reconstructed signal parameters error for f= 250 Hz and SNR= 20 dB for different N values

Lp	N	ΔA [%]	Δf [%]	$\Delta\varphi$ [%]
1	8	5,66	0,471	2,69
2	16	9,51	0,411	2,02
3	32	5,12	0,240	1,07
4	64	4,24	$9{,}57{\cdot}10^{-2}$	0,783
5	128	2,41	0,108	0,749
6	256	1,25	$6{,}84{\cdot}10^{-2}$	0,533
7	512	0,477	$3{,}01{\cdot}10^{-2}$	0,189
8	1024	0,329	$1{,}19{\cdot}10^{-2}$	0,171
9	2048	0,522	$1{,}92{\cdot}10^{-2}$	0,167

Tab. 7. Błędy parametrów zrekonstruowanego sygnału dla f= 250 Hz i N= 2048 dla różnych wartości SNR
Tab. 7. Reconstructed signal parameters error for f= 250 Hz and N= 2048 for different SNR values

Lp	SNR[dB]	ΔA[%]	Δf[%]	$\Delta\varphi$[%]
1	∞	$1{,}47{\cdot}10^{-3}$	$3{,}27{\cdot}10^{-4}$	$1{,}92{\cdot}10^{-3}$
2	40	$5{,}10{\cdot}10^{-2}$	$1{,}86{\cdot}10^{-3}$	$1{,}70{\cdot}10^{-2}$
3	20	0,522	$1{,}92{\cdot}10^{-2}$	0,167
4	10	1,65	$5{,}89{\cdot}10^{-2}$	0,505
5	6	2,64	$9{,}23{\cdot}10^{-2}$	0,791
6	3	3,74	0,129	1,11
7	0	5,28	0,181	1,55

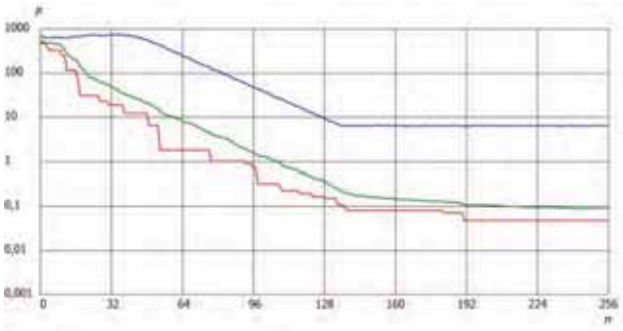
Na podstawie powyższych tabel można zauważyć, że algorytm lepiej radzi sobie z określeniem częstotliwości, niż amplitudy badanego sygnału. Dokładność rekonstrukcji częstotliwości przez badany algorytm jest o rząd wielkości lepsza, niż amplitudy i na ogół nie przekracza kilku procent.

Określenie przyczyn niższej dokładności rekonstrukcji amplitudy wymaga dalszych badań. Wstępne eksperymenty ze zmodyfikowaną wersją badanego algorytmu pokazały, że poprzez wprowadzenie niezależnych współczynników mutacji dla amplitudy i częstotliwości można poprawić dokładność rekonstrukcji amplitudy.

4.2.3. Obserwacja działania algorytmu ewolucyjnego

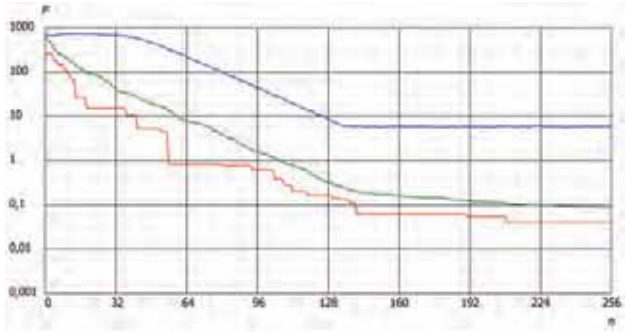
W celu obserwacji działania testowanego algorytmu, program zapisuje dla każdego pokolenia trzy wartości: wartość funkcji przystosowania najlepszego elementu, średnią wartość funkcji przystosowania dla 32 rodziców oraz średnią wartość funkcji przystosowania dla całej populacji. Wartości te zmieniały się w trakcie eksperymentów w zakresie od około 1 na początku eksperymentu do około 10^{-5} dla najlepszego elementu wygenerowanego badany

przez algorytm po kilkuset pokoleniach. Rys. 2 i 3 przedstawiają wykresy tych wartości, przeskalowanych o czynnik 10^3, w skali logarytmicznej, dla dwóch różnych eksperymentów.



Rys. 2. Obserwacja działania algorytmu ewolucyjnego dla *f*=250 Hz i *N*=32 próbek. Kolor niebieski – cała populacja, zielony – populacja rodziców, czerwony – najlepszy element

Fig. 2. The observation of the evolutionary algorithm for *f*=250 Hz and *N*=32 samples. Blue - the entire population, green - the population of parents, red - the best element



Rys. 3. Obserwacja działania algorytmu ewolucyjnego dla *f*=250 Hz i *N*=2048 próbek. Kolory jak na rys. 2

Fig. 3. The observation of the evolutionary algorithm for *f*=250 Hz and *N*=2048 samples. Colors as in fig. 2

Można zauważyć, że średnia wartość funkcji przystosowania całej populacji rośnie podczas pierwszych kilkudziesięciu kroków działania algorytmu. Jest to związane z wysokim początkowym współczynnikiem mutacji, równym 1. Wraz ze spadkiem tego współczynnika wartość ta maleje, po czym ustala się, gdy współczynnik mutacji przyjmuje wartość minimalną równą 0,001.

Wyniki obserwacji zachowania się badanego algorytmu zostaną wykorzystane przy opracowaniu jego bardziej zaawansowanych wersji.

5. Dodatkowe eksperymenty, kierunki dalszych badań

Przeprowadzone dodatkowo próby dotyczyły rozpoznawania sygnału na tle kilku innych sygnałów sinusoidalnych. Testy pokazały, że algorytm ewolucyjny prawie bezbłędnie odnajduje w mieszaninie kilku częstotliwości tę, której amplituda jest najwyższa. Być może umożliwi to dekompozycję złożonego sygnału na jego składowe sinusoidalne. Problem ten będzie przedmiotem dalszych badań.

Wstępnym testom poddana została również zmodyfikowana wersja badanego algorytmu, dla której zastosowano różne współczynniki mutacji dla amplitudy, fazy i częstotliwości. Testy te wykazały, że odpowiedni dobór tych współczynników pozwala zwiększyć dokładność rekonstrukcji badanego sygnału.

6. Podsumowanie

Przeprowadzone badania wykazały, że testowany algorytm ewolucyjny pozwala na dokładną identyfikację parametrów nawet silnie zakłóconego sygnału sinusoidalnego na podstawie niewielkiej liczby losowo pobranych jego próbek. Algorytm ten jest prosty, a przeprowadzone próby z jego zmodyfikowanymi wersjami wskazują, że istnieje możliwość jego dalszego udoskonalania, a także możliwość kontynuacji badań w kierunku rozpoznawania i analizy sygnału o bardziej złożonym kształcie.

W przeciwieństwie do klasycznych metod analizy sygnałów przy pomocy transformaty Fouriera opisywanych w literaturze [2], algorytm ewolucyjny odnajduje z dużą dokładnością częstotliwość i amplitudę sygnału na podstawie niewielkiej liczby jego próbek. Można to wykorzystać w celu analizy i dekompozycji sygnałów dźwiękowych. Wymaga to jednak dalszych badań.

Bibliografia

1. Romberg J., Wakin M., *Compressed Sensing: A Tutorial,* IEEE Statistical Signal Processing Workshop, Madison, Wisconsin 2007.
2. Zieliński T. P., *Cyfrowe przetwarzanie sygnałów,* Wydawnictwa Komunikacji i Łączności, Warszawa 2005.
3. Arabas J., *Wykłady z algorytmów ewolucyjnych,* Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
4. Goldberg D. E., *Algorytmy genetyczne i ich zastosowania,* Wydawnictwa Naukowo-Techniczne, Warszawa, 2003.
5. Yunyun J., Zhen Y., *A distributed compressed sensing approach for speech signal denoising*, "Journal of Electronics (China)", vol. 28, No. 4/5/6 (November 2011), 509–517.
6. Nielsen J. K., Christensen M. G., Jensen S. H., *On compressed sensing and the estimation of continuous parameters from noisy observations,* 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 3609–3612.
7. Hong S., *Direct Spectrum Sensing from Compressed Measurements,* Military Communications conference, 2010 – MILCOM 2010, 1187–1192.
8. Xiao Yu Z., Li C.S., *Compressive Sensing SAR Range Compression with Chirp Scaling Principle,* Science China Information Science, Vol. 55, No. 10/2012, 2292–2300. ■

Reconstruction of noisy sinusoidal signal based on limited number of samples using an evolutionary algorithm

Abstract: The paper presents an experiment involving the use of an evolutionary algorithm to reconstruct the sine wave signal with white noise added based on limited number of its random samples. The experimental results show that such an algorithm is able to identify the test signal with a high accuracy even when the noise level is high (SNR=0 dB). The behavior of the algorithm used is examined, the convergence and accuracy of results are tested, depending on the parameters of the reconstructed signal. The directions for further research are outlined.

Keywords: evolutionary algorithm, signal reconstruction

mgr inż. Piotr Kardasz
Doktorant Wydziału Elektrycznego Politechniki Białostockiej. Zajmuje się badaniami związanych z inteligentnymi algorytmami kompresji, rekonstrukcji i przetwarzania sygnałów.

e-mail: pik@we.pb.edu.pl