Лабораторна робота № 5

Дослідження адаптивних цифрових фільтрів на основі алгоритму LMS

Мета роботи: ознайомлення з адаптивним цифровим фільтром на основі алгоритму LMS, реалізація алгоритму і програми розрахунку адаптивного цифрового фільтру на основі алгоритму LMS, тестування реалізованої програми.

1. Теоретична частина

Серед адаптивних фільтрів найбільшого поширення отримали лінійні адаптивні фільтри з оберненим зв'язком, реалізовані на основі нерекурсивних ЦФ (фільтрів зі скінченною імпульсною характеристикою – СІХ фільтрів). Структурна схема такого адаптивного фільтра зображена на рис. 5.1.

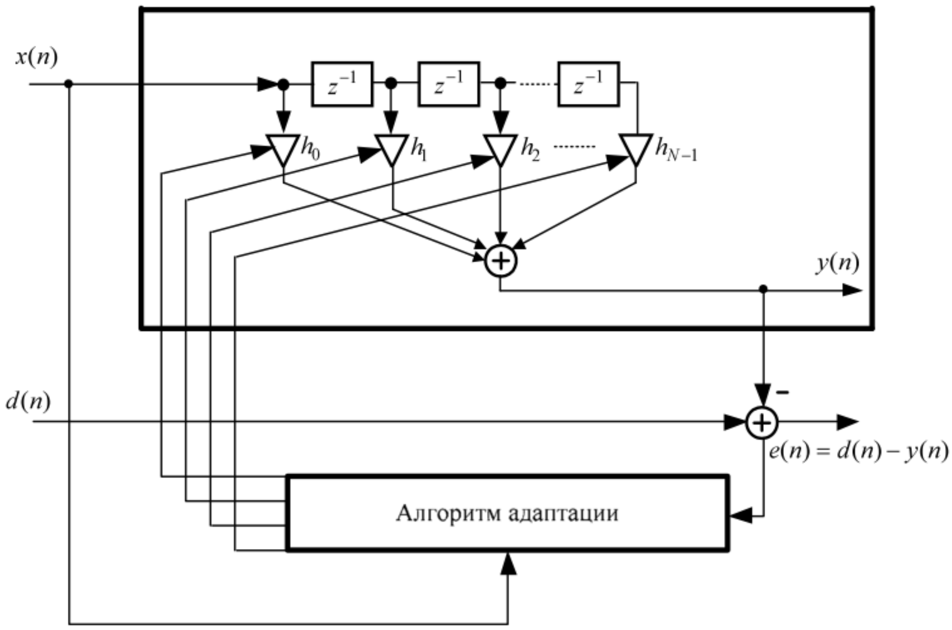

Рис. 5.1. Структурна схема лінійного адаптивного фільтра з оберненим зв'язком

На виході адаптивного фільтра формуються два сигнали:

- вихідний сигнал y_k;
- сигнал похибки e_k:

$$e_k = d_k - y_k\,. \tag{5.1}$$

Згідно різницевого рівняння вихідний сигнал СІХ – фільтра рівний лінійній комбінації відліків вхідного сигналу:

$$y_k = \sum_{i=0}^{N-1} h_i x_{k-i}\,, \tag{5.2}$$

де $h_i, i = 0,1,...,N-1$ - коефіцієнти СІХ – фільтра. Коефіцієнти різницевого рівняння СІХ – фільтра співпадають з відліками його імпульсної характеристики.

Згідно (5.2) сигнал похибки рівний:

$$e_k = d_k - \sum_{i=0}^{N-1} h_i x_{k-i} \ . \qquad (5.3)$$

В методі найменших квадратів (МНК) – алгоритмі LMS (Least Mean Squares) і його модифікації НМНК (нормований метод найменших квадратів) реалізовано рекурентне обчислення оцінок параметрів адаптивного фільтра $\hat{h}_k$. Як критерій найкращого наближення вихідного сигналу y_k до взірцевого сигналу d_k використовується мінімум квадрату сигналу похибки:

$$e_k^2 = \left[d_k - \sum_{i=0}^{N-1} h_i x_{k-i} \right]^2 \to \min_h . \qquad (5.4)$$

Для цього адаптивного фільтра оптимізуюча функція є функцією часу k:

$$F(k,h) = e_k^2 = \left[d_k - \sum_{i=0}^{N-1} h_i x_{k-i} \right]^2 . \qquad (5.5)$$

Для пошуку мінімуму функції $F(k,h)$ використовується градієнтний метод найшвидшого спуску – ітераційна процедура, яка визначає траєкторію найшвидшого покрокового наближення до мінімуму, де крокам ітерації відповідають моменти дискретного нормованого часу k.

У відповідності з даним методом, на кожному кроці ітерації оцінюється вектор $\hat{h}_{k+1}$, зміщений відносно вектора $\hat{h}_k$ на величину, пропорційну градієнту функції $F(k,h)$ в точці k:

$$\hat{h}_{k+1} = \hat{h}_k + \frac{\mu}{2} \nabla_k , \qquad (5.6)$$

де $\hat{h}_{k+1}$ - вектор оцінок параметрів адаптивного фільтра в момент часу $k+1$; μ - додатна константа (крок адаптації); ∇_k - градієнт функції $F(k,h)$, що визначається як вектор, елементами якого є частинні похідні даної функції по всіх h_i в момент часу k:

$$\nabla_k = 2\left[d_k - \sum_{i=0}^{N-1} h_i x_{k-i} \right] x_{k-i} \qquad (5.7)$$

або з врахуванням (5.3):

$$\nabla_k = 2 e_k x_{k-i} , \ \ i = 0,1,...,N-1. \qquad (5.8)$$

Після підстановки (5.8) в (5.6) отримуємо рекурентну формулу для оцінок параметрів адаптивного фільтра:

$$\hat{h}_{k+1} = \hat{h}_k + \mu e_k x_k , \qquad (5.9)$$

де x_k - вектор відліків вхідного сигналу.

Значення кроку адаптації μ впливає на швидкість збіжності оцінок параметрів адаптивного фільтра до оптимальних параметрів h_{opt} в фільтрі Вінера.

При виборі значення μ з діапазону

$$0 < \mu < 2/\lambda_{max}, \tag{5.10}$$

де λ_{max} - максимальне власне значення матриці R_x, гарантується збіжність в середньому – для середніх значень оцінок параметрів адаптивного фільтра при $k \to \infty$.

Практичне значення має діапазон значень кроку адаптації μ, що гарантує збіжність в середньому квадраті – для середніх квадратів оцінок параметрів адаптивного фільтра, при умові, що останні прямують до фіксованих значень. Це забезпечується для значень μ в діапазоні

$$0 < \mu \le 2/N \cdot P_x, \tag{5.11}$$

де P_x - середній квадрат вхідного сигналу x_k довжиною L:

$$P_x = \frac{1}{L}\sum_{k=0}^{L-1} x_k^2. \tag{5.12}$$

Значення кроку адаптації μ вибирається з компромісних міркувань: з одного боку воно впливає на швидкість збіжності алгоритму LMS (чим більше μ, тим більше він відрізняється від сигналу похибки в фільтрі Вінера).

В нормалізованому алгоритмі NLMS рекурентна формула (5.9) замінюється іншою, в якій крок адаптації μ залежить від часу:

$$\hat{h}_{k+1} = \hat{h}_k + \mu_k e_k x_k, \tag{5.13}$$

так як він нормується до енергії сигналу x_{k-i}, $i = 0,1,...,N-1$:

$$\mu_k = \frac{\mu_0}{x_k^t x_k + \varepsilon}, \tag{5.14}$$

де $x_k^t x_k$ - енергія сигналу, рівна добутку вектора – рядка x_k^t на вектор – стовпець x_k; μ_0 - фіксоване значення кроку, яке впливає на збіжність алгоритму адаптації (вибирається з діапазону $0 < \mu_0 < 2$); ε- мала додатна константа, яка визначає максимальне значення μ_k, рівне μ_0/ε при нульовому вхідному сигналі.

Ітераційна процедура обчислення оцінок параметрів адаптивного фільтра в алгоритмах LMS і NLMS включає в себе наступні кроки:

1. $k = 0$. Задання початкових (зазвичай нульових) значень оцінок параметрів адаптивного фільтра $h(0)$.

2. Обчислення вихідного сигналу y_k згідно виразу (5.2).

3. Обчислення сигналу похибки e_k згідно виразу (5.3).

4. Оновлення оцінок параметрів адаптивного фільтра $\hat{h}_{k+1}$ згідно виразу (5.9) або (5.13).

5. $k = k+1$. Перехід до нової ітерації.

5. Повторення кроків 2 – 5.

Основною перевагою алгоритму LMS є його простота, а недоліком відносно повільна збіжність ітераційної процедури обчислення параметрів адаптивного фільтра.

Завдання до лабораторної роботи

Вихідні дані для завдання цієї лабораторної роботи приведені в табл. 5.1.

Табл. 5.1. Вихідні дані для проектування адаптивного цифрового фільтра на основі алгоритму LMS

Змінна	Призначення	Значення	Ідентифікатор
$N_{var.}$	Номер варіанту	$N_{var.}$	N_v
N	Довжина СІХ фільтра в складі АЦФ	$N=\left(N_{var.}\ mod\ 10\right)+41$	N
L	Довжина вибірки вхідного сигналу	$L=500\left(N_{var.}\ mod\ 10\right)+100$	L
$f_{s.}$	Частота дискретизації	$f_{s.}=2000\left(N_{var.}\ mod\ 10\right)$	F_s
A_1	Амплітуда сигналу	$A_1=0.25+0.01N_{var.}$	A_1
f_1	Частота сигналу	$f_1=f_{ä.}/8$	f_1

1. *Моделювання нормального білого шуму*. Створити модель нормального білого шуму $r\left(n\right)$ (ідентифікатор r_gauss) довжини L з нульовим середнім значенням і одиничною дисперсією.

2. Моделювання структури АЦФ з алгоритмом LMS. Створити модель структури АЦФ з алгоритмом LMS у вигляді об’єкта з іменем AFIR_LMS і вивести його властивості.

Задати вхідні параметри об’єкта:

- довжину СІХ – фільтра N;

- крок адаптації μ (ідентифікатор m_u), рівний половині максимального кроку адаптації (ідентифікатор mu_max) в (5.11).

При обчисленні кроку адаптації μ як вхідний сигнал АЦФ $x\left(n\right)$ (ідентифікатор x) використовувати нормальний білий шум $r\left(n\right)$: $x\left(n\right)=r\left(n\right)$; решта параметрів вибираються по замовчуванню.

3. Моделювання структури АЦФ з алгоритмом NLMS. Створити модель структури АЦФ з алгоритмом NLMS у вигляді об’єкта з іменем AFIR_NLMS і вивести його властивості.

Задати вхідні параметри об’єкта:

- довжину СІХ – фільтра N;

- константу $\varepsilon=10^{-6}$ (ідентифікатор epsilon) в (5.14) ;

- решта параметрів вбираються по замовчуванню.

4. Фільтрація сигналу від шумів з допомогою АЦФ.

ЛАБОРАТОРНИЙ ПРАКТИКУМ
АДАПТИВНІ СИСТЕМИ ОПРАЦЮВАННЯ ІНФОРМАЦІЇ

Моделювання процесу фільтрації сигналу від шуму включає в себе наступні кроки:

- моделювання вхідного сигналу для невідомого АЦФ $x(n)=x_{\o}(n)$ (ідентифікатор x). Як шум (ідентифікатор x_noise) вибрати нормальний білий шум $r(n)$: $x_{\o}(n)=r(n)$.

- моделювання корисного сигналу $s(n)$ (ідентифікатор s) у вигляді періодичної послідовності з періодом L:

$$s(n)=A_1 cos(2\pi f_1 nT). \tag{5.15}$$

Тотожне перетворення останнього виразу:

$$s(n)=A_1 cos\left(\frac{2\pi f_1}{f_{\ddot{a}}}n\right)=A_1 cos(\hat{\omega}_1 n). \tag{5.16}$$

- моделювання вхідного сигналу АЦФ $d(n)$ (ідентифікатор d) у вигляді адитивної суміші сигналу $s(n)$ з шумом $x_{\o}(n)$:

$$d(n)=s(n)+x_{\o}(n). \tag{5.17}$$

- обчислення вихідного сигналу $y(n)$ і сигналу похибки $\varepsilon(n)$ АЦФ з іменем AFIR_LMS (ідентифікатори y_lms і e_lms).

Вивести графіки:

- корисного сигналу $s(n)$ і його адитивної суміші з шумом $d(n)$;
- сигналів похибки АЦФ (оцінок корисного сигналу) для об’єктів AFIR_LMS і AFIR_NLMS;
- оцінок корисного сигналу в усталеному режимі з використанням алгоритмів LMS і NLMS.

Початковий момент усталеного режиму $n_{\ddot{\i}\hat{\i}\div.}$ (ідентифікатор n_start) задати рівним $0.05L$.

Додаток 1. Фрагмент програми генерації масиву випадкових чисел з нормальним розподілом.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Period parameters */
#define N 624
#define M 397
#define MATRIX_A 0x9908b0df   /* constant vector a */
#define UPPER_MASK 0x80000000 /* most significant w-r bits */
#define LOWER_MASK 0x7fffffff /* least significant r bits */

/* Tempering parameters */
```

```
#define TEMPERING_MASK_B 0x9d2c5680
#define TEMPERING_MASK_C 0xefc60000
#define TEMPERING_SHIFT_U(y)  (y >> 11)
#define TEMPERING_SHIFT_S(y)  (y << 7)
#define TEMPERING_SHIFT_T(y)  (y << 15)
#define TEMPERING_SHIFT_L(y)  (y >> 18)

#define PI 3.1415926

static unsigned long mt[N]; /* the array for the state vector  */
static int mti=N+1; /* mti==N+1 means mt[N] is not initialized */

/* initializing the array with a NONZERO seed */
void
sgenrand(unsigned long seed)

{
   /* setting initial seeds to mt[N] using         */
   /* the generator Line 25 of Table 1 in          */
   /* [KNUTH 1981, The Art of Computer Programming */
   /*    Vol. 2 (2nd Ed.), pp102]                  */
   mt[0]= seed & 0xffffffff;
   for (mti=1; mti<N; mti++)
      mt[mti] = (69069 * mt[mti-1]) & 0xffffffff;
}

double /* generating reals */
/* unsigned long */ /* for integer generation */
genrand()
{
   unsigned long y;
   static unsigned long mag01[2]={0x0, MATRIX_A};
   /* mag01[x] = x * MATRIX_A  for x=0,1 */

   if (mti >= N) { /* generate N words at one time */
      int kk;

      if (mti == N+1)   /* if sgenrand() has not been called, */
         sgenrand(4357); /* a default initial seed is used   */

      for (kk=0;kk<N-M;kk++) {
         y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
         mt[kk] = mt[kk+M] ^ (y >> 1) ^ mag01[y & 0x1];
      }
```

```
        for (;kk<N-1;kk++) {
            y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >> 1) ^ mag01[y & 0x1];
        }
        y = (mt[N-1]&UPPER_MASK)|(mt[0]&LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >> 1) ^ mag01[y & 0x1];

        mti = 0;
    }

    y = mt[mti++];
    y ^= TEMPERING_SHIFT_U(y);
    y ^= TEMPERING_SHIFT_S(y) & TEMPERING_MASK_B;
    y ^= TEMPERING_SHIFT_T(y) & TEMPERING_MASK_C;
    y ^= TEMPERING_SHIFT_L(y);

    return ( (double)y / (unsigned long)0xffffffff ); /* reals */
    /* return y; */ /* for integer generation */
}

int main()
{  int i, k, n_1 = 1000;
   double y_1[2000];
   double xround_1, xround_2;


   for (i = 1; i <= n_1; ++i)
       {
Lb_1:    xround_1 = genrand();
         xround_2 = genrand();
         y_1[i]   = sqrt(-2.0*log(xround_1))*cos(2.0*PI*xround_2);
         if (abs(y_1[i]) >= 10.0) { goto Lb_1; }
       }
    FILE *filePR;
    filePR = fopen("Wait_noise.txt","w");
    for (k = 1; k <= n_1; ++k) fprintf(filePR," %11.5f\n", y_1[k]);
    fclose(filePR);

    return 0;
}
```