Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Кафедра РТС

Отчет по лабораторной работе №4

«ИССЛЕДОВАНИЕ СВОЙСТВ КОДА РИДА – СОЛОМОНА»

Выполнил:

ст.гр.240102
shlom41k

Проверил:

Панькова В.В.

Минск 2015

Цель работы

1. Изучить методы помехоустойчивого кодирования и декодирования информации с помощью циклических блочных кодов Рида – Соломона.
2. Исследовать свойства кодов Рида – Соломона (РС) во временной и спектральной областях.
3. Изучить связь корректирующей способности кодов с конечными полями и спектральными преобразованиями Фурье – Галуа.
4. Приобрести навыки построения алгоритмов и программ кодирования и декодирования информации.

Выполнение работы

Кодирование РС кодом во временной области

```
> restart:
> with(linalg):
  with(plots):
```

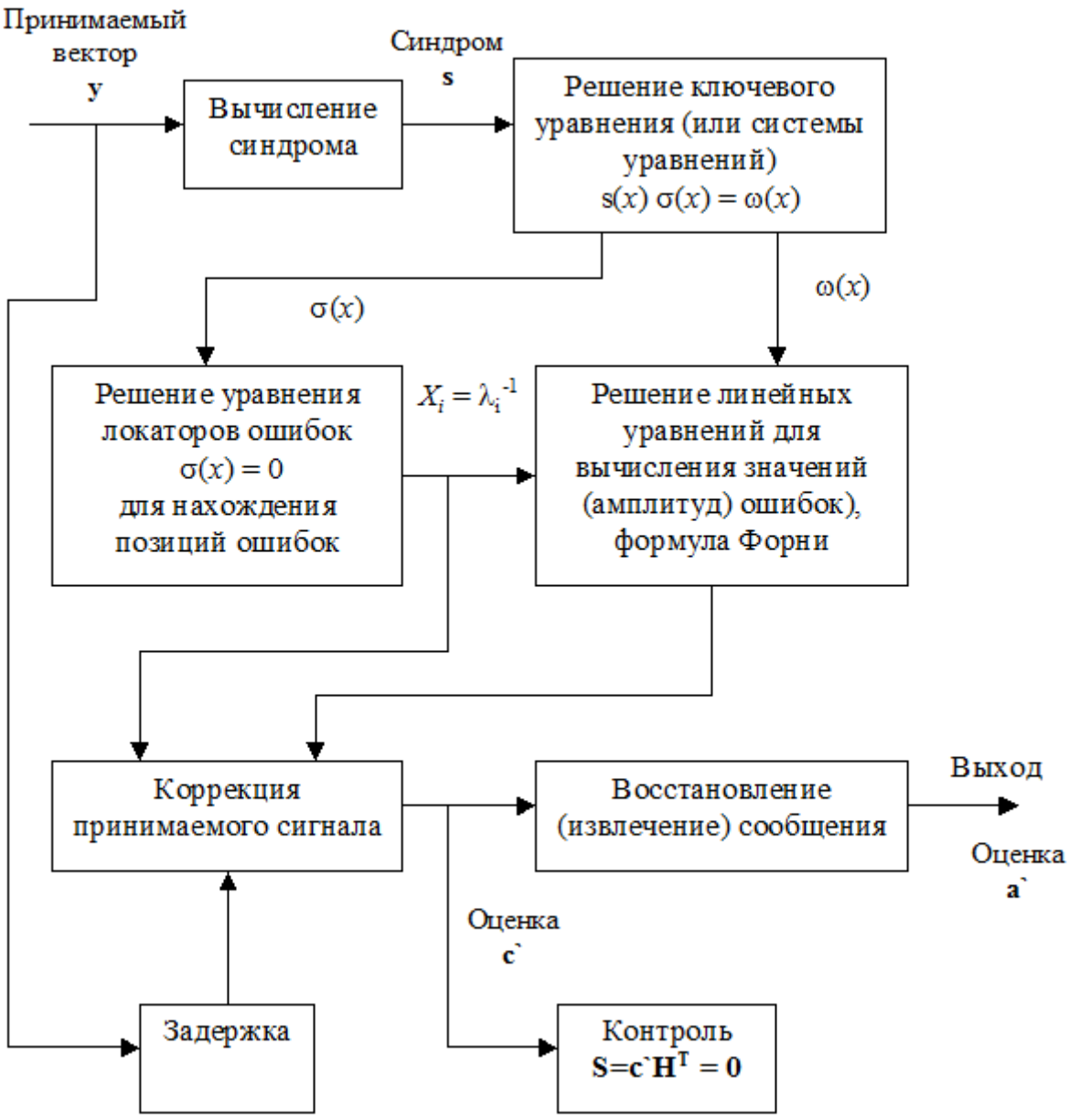


Рисунок *1* – Схема алгебраического декодера РС во временной области

Вычисление произведения вектора v на матрицу A, vA по модулю p(mod p)

```
> read"VMP.m";
```

Вычисление веса Хэмминга вектора w

```
> read"HamWeight.m";
```

Вычисление расстояния Хэмминга d(u,v) двух векторов

```
> read"HamDistance.m";
```

Преобразование вектор - полином с коэффициентами = элементам вектора

```
> read"ToPoly.m";
```

Конвертирование полинома в вектор (длины n) n должно быть >= degree(poly,var) + 1

```
> read "ToVector.m";
```

Выбор неприводимого полинома степени r пад полем F_p путем факторизации полинома x^(p^r-1)-1 mod p.

```
> read "IrredPoly.m";
```

Вычисление минимального полинома элемента beta^s, где beta = RootOf(h) - примитивный элемент поля GF(p^r), где r = deg(h). Полином h должен быть неприводимым над полем GF(p) и выполняться неравенство 1 <= s <= 2^r-2

```
> read "MinPoly.m";
```

Нахождение генераторного полинома кода РС, "irred" - должен быть неприводимым полиномом степени r над полем F_p, корни которого являются примитивными элементами для поля GF(p^r). Корнями генераторного полинома являются элементы beta^(m+1), ... , beta^(m+delta-1)

```
> read "RSGenPoly.m";
```

Полиномиальная форма кодирования информации кодом РС . "delta" - минимальное кодовое расстояние; delta=degree(gen)+1. Длина информационного слова должна быть равна p^r-delta. Переменная "var" должна совпадать с символикой переменной генераторного полинома

```
> read "RSEncode.m";
```

Вычисление синдрома ошибки принятого вектора. Результат представляется в виде полинома переменной *x*.

```
> read "RSBMSynPol.m";
```

Вычисление полинома локаторов ошибок по алгоритму Берлекэмпа-Месси. Мин. кодовое расстояние "delta" - нечетное число."irred " - неприводимый полином степени r , соответствующий степени расширенного поля GF(p^r). Синдром "syndromepoly" должен иметь форму полинома относительно переменной x

```
> read "RSBMErrorLoc.m";
```

Вычисление местоположения ошибок и значений ошибок кода РС (p^r,delta)

```
> read"RSBMErrVal.m";
```

Алгоритм общего декодирования кода РС с восстановлением кодового слова

```
> read "RSBMDecode.m";
```

Модель канала

```
> read "GHerr.m";
```

```
> read "Dganal.m";read "ToGraf.m";
```

Пример

Задание параметров кода

Код РС **(n=31,d=5)** над полем **GF(2^5=32) ,p=2, r=5**

```
> n:=31;d:=9;p:=2;r:=5;   ki:=n-d+1;
```

1. Определим неприводимый полином

```
> ir:=IrredPoly(r,p);
```

2.Вычислим генераторный полином кода РС. Обозначим корень примитивный элемент поля **"beta"** как корень неприводимого полинома.

```
> alias(beta=RootOf(ir));
> g:=RSGenPoly(p,r,d,0,ir,x);
```

3. Зададим информационный вектор **a**

3.1. Сформируем случайный вектор **h** с помощью оператора формирования полинома значности **ki** и коэффициентами из элементов поля **GF**

```
> h:=Randpoly(ki-1,x,beta) mod 2;
```

3.2. Определим множество **a** из **ki** элементов

```
> a:=array(1..ki);
```

3.3. Преобразуем полином **h** в вектор **a**

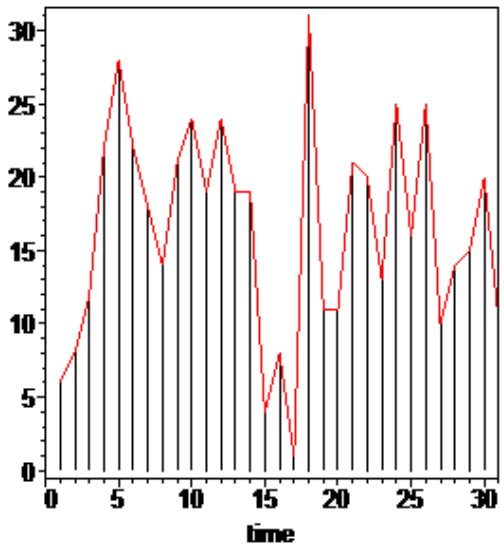
```
> a:=ToVector(h,x,ki,2);
```

4. Вычислим кодовое слово c(x), представленное в полиномиальной форме

```
> c:=RSEncode(a,g,x,ir,2);
```

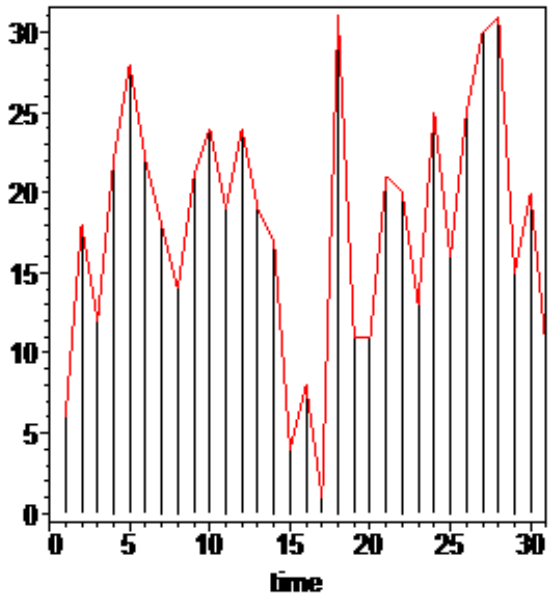
Отображение

```
> cp:=Dganal(c,ir,n,r,beta):
> pc:=[shseq([i,cp[i]],i=1..n)]:
> pic1:=loplot(pc,time=0..n,axes=boxed,xtickmarks=n):
> p2:=mcarray([seq(cp[j1], j1=1..n)]):
> pic2:=ikToGraf(p2,n):
> display(pic1,pic2);
```



5. Моделирование канала передачи информации: с помощью оператора **GHerr** сформируем вектор **y = c + e mod p** - аддитивная смесь сигнала **c** и ошибки **e**

```
> #read "GHerr.m";
> dir:=ir;
> y:=GHerr(c,3,dir,n,2,beta);
> cp2:=Dganal(y,ir,n,r,beta);
> pc2:=[seq([i,cp2[i]],i=1..n)]:
> pic12:=plot(pc2,time=0..n,axes=boxed,xtickmarks=n):
> p22:=array([seq(cp2[j1], j1=1..n)]):
> pic22:=ToGraf(p22,n):
> display(pic12,pic22);
```



6. Вычислим полином синдрома ошибок **s(x)**

```
> s:=RSBMSynPol(y,d,0,ir,2);
```

7. Вычислим полином локаторов ошибок **q(x)** с помощью Алгоритма Берлекемпа-Месси (Рисунок 2).

```
> q:=RSBMErrorLoc(s,d,ir,2,i=l);
```

8. Вычислим оценку вектора ошибок **E** (Рисунок 3)**.**

```
> E:=RSBMErrVal(q,s,0,d,ir,2);
```

9. Выполним полное декодирование и восстановление кодового слова

```
> ce:=RSBMDecode(y,2,ir,d,0);
```

Отображение

```
> cp3:=Dganal(ce,ir,n,r,beta):
> pc3:=[seq([i,cp3[i]],i=1..n)]:
> pic3:=plot(pc3,time=0..n,axes=boxed,xtickmarks=n):
> p23:=array([seq(cp3[j1], j1=1..n)]):
> pic4:=ToGraf(p23,n):
> display(pic3,pic4);
```

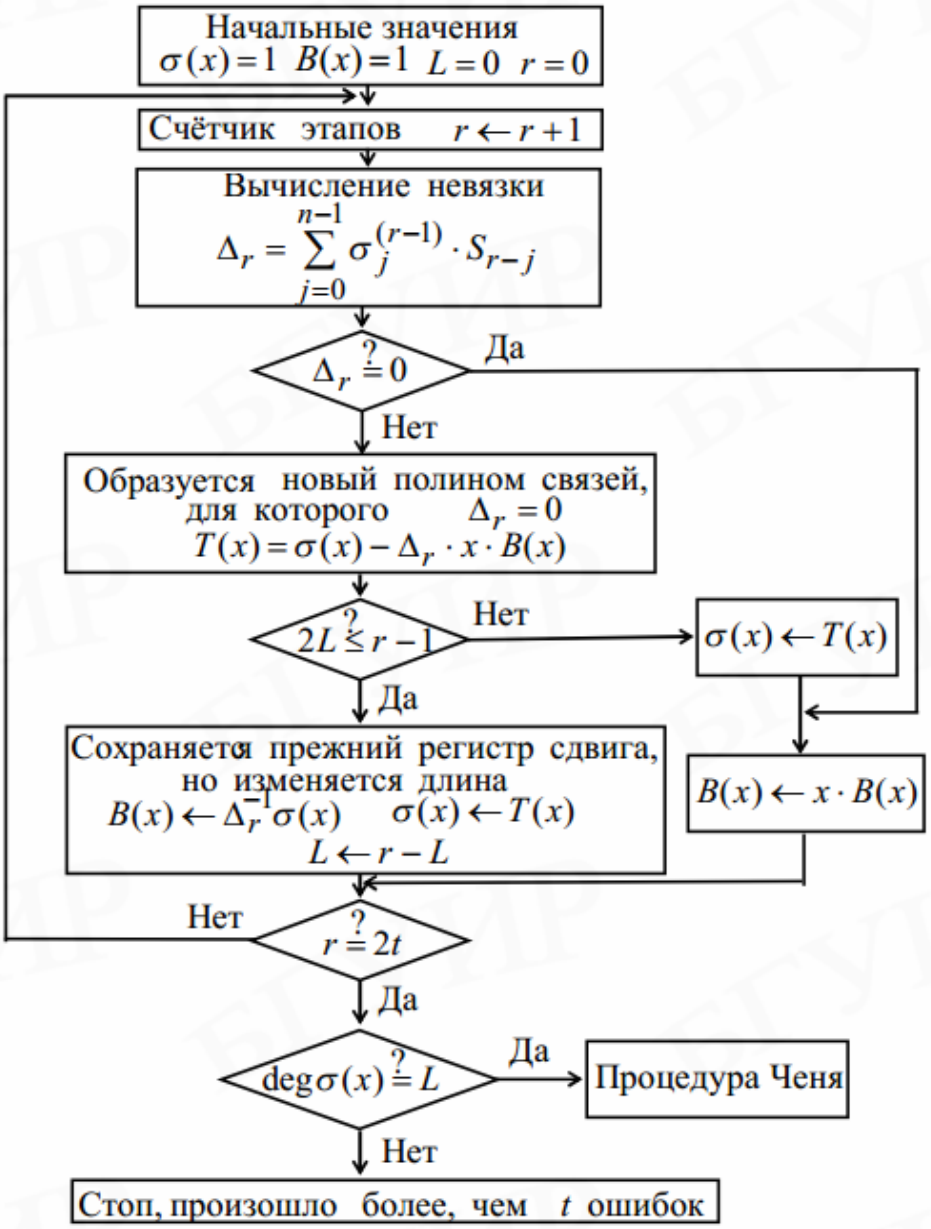


Рисунок *2* – Алгоритм Берлекемпа-Месси

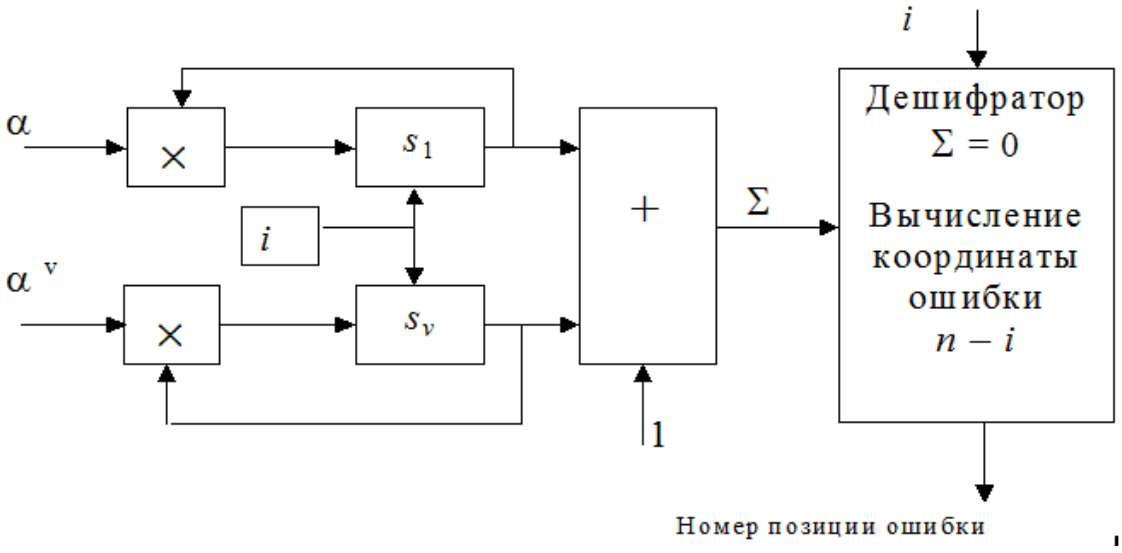
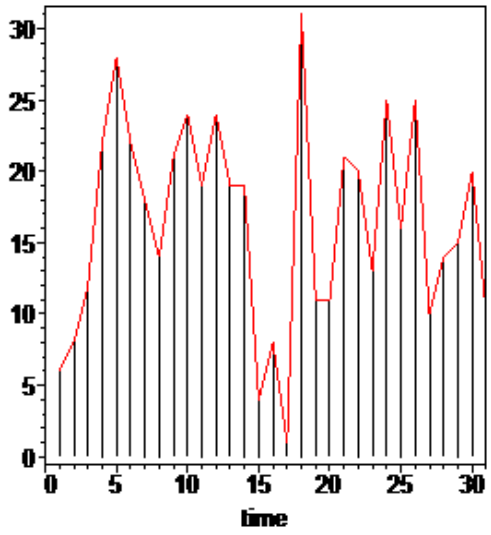


Рисунок *3* – Алгоритм Ченя



10. Проверка равенства переданного вектора **c** и векторов оценки **ce.** Вычислим расстояние Хэмминга между двумя векторами c и ce. Если эти вектора равны, то расстояние Хэмминга равно 0.

```
> HamDistance(c,ce);
```

Кодирование РС кодом в частотной области

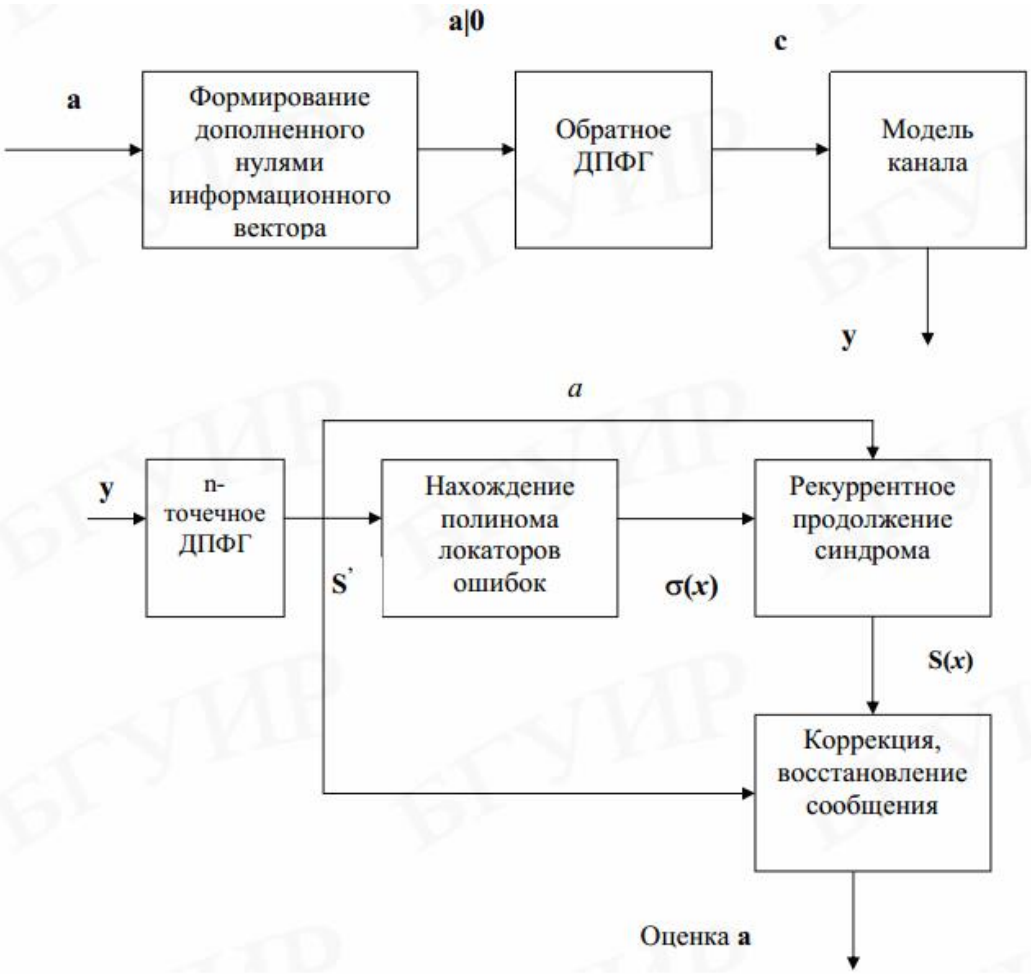


Рисунок *4* – Схема спектрального кодера-декодера:
а) кодирование; б) декодирование

```
> restart:
```

Подключение библиотеки лин. алгебры

```
> with(linalg):
```

Формирование конечного поля F(q), q=p^m, m=1

```
> p:=17;
> m:=1;
> q:=p^m;
> powers:=vector(q-1);
> alpha:=3;
> for i to q-1 do powers[i]:=evala(alpha^(i-1)) mod p od:
> print(powers);
> inverse:=vector(q-1);
> for i to q-1 do inverse[i]:=evala(1/powers[i])mod p od:
> print(inverse);
```

Кодирование

```
> k:=8;
```

Информационный вектор

```
> fin:=vector(q-1,[1, 1, 1, alpha, 1+alpha, 1, alpha^2, 0, 0, 0, 0 , 0, 0, 0, 0, 0]);
```

Прямое преобразование Фурье

Задание выражения для прямого преобразования Фурье через вычисление суммы

```
> fx:= x-> simplify(add(fin[i]*x^(i-1), i=1..k)) mod p;
```

Постановка значений сигнала в формулу для вычисления ПФГ и отображение результата вычислений

Формирование кодового слова w

```
> w:=map(fx, powers);
```

Обратное преобразование фурье

Задание выражения для обратного преобразования Фурье через вычисление суммы

```
> fx1:= x-> simplify((1/16)*add(fin[i]*x^(-(i-1)), i=1..k)) mod p;
```

Постановка значений сигнала в формулу для вычисления ПФГ и отображение результата вычислений

```
> w1:=map(fx1, powers);
```

Прямое матричное преобразование Фурье

Задание массива элементов памяти

```
> FT:=matrix(q-1,q-1,[]);
```

Формирование матрицы прямого ПФГ

```
> for r to q-1 do
  for s to q-1 do
     FT[r,s]:=simplify(alpha^((r-1)*(s-1))) mod p
  od:od: print(FT);
```

Умножение матрицы на вектор- матричная форма ПФГ

```
> w:=evalm(FT&*fin);
```

Нормализация сигнала по модулю p

```
> w:=map(`mod`,w,p);
```

Обратное матричное преобразование Фурье

Задание массива элементов памяти

```
> FT1:=matrix(q-1,q-1,[]); 1/16 mod 17;
```

Формирование матрицы обратного ПФГ

```
> for r to q-1 do
  for s to q-1 do
     FT1[r,s]:=simplify((1/16)*alpha^(-(r-1)*(s-1))) mod p
  od:od: print(FT1);
```

Умножение матрицы на вектор- матричная форма ОПФГ

```
> w1:=evalm(FT1&*w);
```

Нормализация сигнала по модулю p

```
> w1:=map(`mod`,w1,p);
```

Декодирование с помощью обратного преобразования

```
> w3:=evalm(FT1&*w);
> w3:=map(`mod`,w3,p);
```

Модель канала передачи информации

Вектор ошибок.

```
> ee:=vector(q-1,[0, 1, 0, 1+alpha, 0, 0, 0, 0, 1, 0, 0, 0, 0, alpha,0,0]);
```

Аддитивная смесь сигнала и помехи

```
> rr:=evalm(ee+w):
```

Нормализация суммы по модулю p

```
> rr:=map(`mod`,rr,p);
```

Первый этап декодирования - вычисление преобразования Фурье

Вычисление спектра принятого сигнала

```
> R:=evalm(FT1&*rr);
> R:=map(`mod`,R,p);
```

Селекция из спектра компонент синдрома

```
> R[15];
```

Вывод

В данной лабораторной работе были изучены методы помехоустойчивого декодирования информации с помощью циклических блочных кодов Рида – Соломона. Исследовали свойства кодов Рида – Соломона во временной и спектральных областях. Приобрели навыки построения алгоритмов и программ кодирования и декодирования информации.