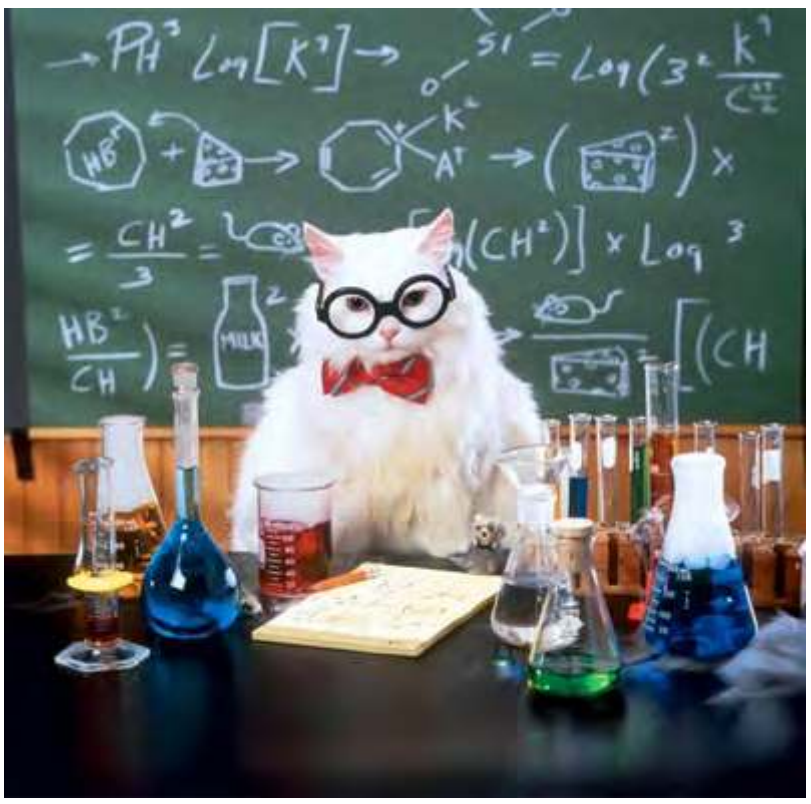
akuzmin 26 августа 2013 в 18:30

Коды Рида-Соломона. Простой пример

Алгоритмы

Из песочницы



Благодаря кодам Рида-Соломона можно прочитать компакт-диск с множеством царапин, либо передать информацию в условиях связи с большим количеством помех. В среднем для компакт-диска избыточность кода (т.е. количество дополнительных символов, благодаря которым информацию можно восстанавливать) составляет примерно 25%. Восстановить при этом можно количество данных, равное половине избыточных. Если емкость диска 700 Мб, то, получается, теоретически можно восстановить до 87,5 Мб из 700. При этом нам не обязательно знать, какой именно символ передан с ошибкой. Также стоит отметить, что вместе с кодированием используется перемежевание, когда байты разных блоков перемешиваются в определенном порядке, что в результате позволяет читать диски с обширными повреждениями, локализированными близко друг к другу (например, глубокие царапины), так как после операции, обратной перемежеванию, обширное повреждение оборачивается единичными ошибками во множестве блоков кода, которые поддаются восстановлению.

Давайте возьмем простой пример и попробуем пройти весь путь – от кодирования до получения исходных данных на приемнике. Пусть нам нужно передать кодовое слово C, состоящее из двух чисел – 3 и 1 именно в такой последовательности, т.е. нам нужно передать вектор C=(3,1). Допустим, мы хотим исправить максимум две ошибки, не зная точно, где они могут появиться. Для этого нужно взять 2*2=4 избыточных символа. Запишем их нулями в нашем слове, т.е. C теперь равно (3,1,0,0,0,0). Далее необходимо немного разобраться с математическими особенностями.

Поля Галуа

Многие знают романтическую историю о молодом человеке, который прожил всего 20 лет и однажды ночью написал свою математическую теорию, а утром был убит на дуэли. Это Эварист Галуа. Также он несколько раз пытался поступить в университеты, однако экзаменаторы не понимали его решений, и он проваливал экзамены. Приходилось ему учиться самостоятельно. Ни Гаусс, ни Пуассон, которым он послал свои работы, также не поняли их, однако его теория отлично пригодилась в 60-х годах XX-го века, и активно используется в наше время как для теоретических вычислений в новых разделах математики, так и на практике.

Мы будем использовать довольно простые выводы, следующие из его теории групп. Основная идея – есть конечное (а не бесконечное) количество чисел, называемое полем, с помощью которых можно проводить все известные математические операции. Количество чисел в поле должно являться простым числом в любой натуральной степени, однако в случае простых кодов Рида-Соломона, рассматриваемых здесь, размерность поля — простое число в степени 1. В расширенных кодах Рида-Соломона степень более 1.
Например, для поля Галуа размерности 7, т.е. GF(7), все математические операции будут происходить с числами 0,1,2,3,4,5,6.
Пример сложения: 1+2=3; 4+5=9 mod 7=2. Сложение в полях Галуа представляет собой сложение по модулю. Вычитание и умножение также делается по модулю.
Пример деления: 5/6 = 30/36 = 30/(36 mod 7) = 30/1 = 30 = 30 mod 7 = 2.
Возведение степень происходит аналогично умножению.

z	z^0	z^1	z^2	z^3	z^4	z^5	z^6
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	1	2	4	1	2	4	1
3	1	3	2	6	4	5	1
4	1	4	2	1	4	2	1
5	**1**	**5**	**4**	**6**	**2**	**3**	**1**
6	1	6	1	6	1	6	1

Полезное свойство обнаруживается в полях Галуа при возведении в степень. Как можно заметить, если возвести в степень числа 3 либо 5 в выбранном поле Галуа GF(7), в строке присутствуют все элементы текущего поля Галуа кроме 0. Такие числа называются примитивными элементами. В кодах Рида-Соломона обычно используется самый большой примитивный элемент выбранного поля Галуа. Для GF(7) он равен 5.

Можно отметить, что числа в полях Галуа – это вместе с тем и абстракции, которые имеют более тесную связь между собой, чем привычные нам числа.

Интерполяция

Как многие знают еще со школьного курса, интерполяция – это нахождение многочлена минимальной степени в данном наборе точек. Например, есть три точки и три значения какой-то функции в этих точках. Можно найти функцию, которая удовлетворяет таким входным данные. Например, это очень просто сделать с помощью преобразования Лагранжа. После того, как функция найдена, можно построить еще несколько точек, и эти точки будут связаны с тремя исходными точками. Формирование избыточных символов при кодировании – это операция, похожая на интерполяцию.

Обратное дискретное преобразование Фурье (IDFT)

Преобразование Фурье, наряду с теорией групп Галуа – это еще одна вершина математической мысли, которая в сегодняшнее время используется во множестве областей. Некоторые даже считают, что преобразование Фурье описывает один из фундаментальных законов Вселенной. Основная суть: любой непериодический сигнал конечной протяженности можно представить в виде суммы синусоид разных частот и фаз, потом по ним можно заново построить исходный сигнал. Однако это не единственное описание. Числа в полях Галуа напоминают упомянутые разные частоты синусоид, так что преобразование Фурье можно использовать и для них.
Дискретное преобразование Фурье – это формула преобразования Фурье для дискретных значений. Преобразование имеет два направления – прямое и обратное. Обратное преобразование проще математически, поэтому давайте закодируем рассматриваемое слово C=(3,1,0,0,0,0) с помощью него. Первые два символа – информация, последние четыре – избыточные и всегда равны 0.
Запишем кодовое слово C в виде полинома: $C=3*x^0+1*x^1 + 0*x^2 +\ldots = 3+x$. Как поле Галуа возьмем упомянутое GF(7), где примитивный элемент = 5. Делая IDFT, находятся значения полинома C для примитивного элемента z разных степеней. Формула IDFT: $c_j=C(z^j)$. То есть, находим значения функции $C(z^j)$, где j – элементы поля Галуа GF(7). Мы рассчитываем до j=N-2=7-2=5 степени. Посмотрев на таблицу степеней, можно догадаться, почему: в шестой степени значение снова 1, т.е. повторяется, вследствие чего потом невозможно было бы определить, в какую степень возводили – в 6ю или 0ю.
$c_0 = C(z^0) = 3 + 1* z^0 = 3 + 1* 5^0 = 4$
$c_1 = C(z^1) = 3 + 1* z^1 = 8 = 1$
$c_2 = C(z^2) = 3 + 1* z^2 = 0$
…
Таким образом C(3,1,0,0,0,0) => c(4,1,0,2,5,6).

Мы передаем слово c(4,1,0,2,5,6).

Ошибка

Ошибка представляет собой еще одно слово, которое суммируется с передаваемым. Например, ошибка f=(0,0,0,2,0,6). Если сделать c + f, то получим c_f(4,1,0,4,5,5).

Прямое преобразование Фурье (DFT). Декодирование. Синдром

На приемнике мы получили слово c+f = c_f(4,1,0,4,5,5). Как проверить, были ли ошибки при передаче? Известно, что мы кодировали информацию с помощью IDFT в GF(7). DFT (дискретное преобразование Фурье) – это преобразование, обратное IDFT. Проделав его, можно получить исходную информацию и четыре нуля (т.е. C(3,1,0,0,0,0)), в случае, если ошибок не было. Если же ошибки были, то вместо этих четырех нулей будут другие цифры. Проделаем DFT для c_f и проверим, есть ли ошибки. Формула DFT: $C_k=N^{-1}*c(z^{-kj})$. По-прежнему используется примитивный элемент z=5 поля GF(7).
$C_0 = c(5^{-0*j})/6 = (4*5^{-0*0} + 1*5^{-1*0} + 0*5^{-2*0} + 4*5^{-3*0} + 5*5^{-4*0} + 5*5^{-5*0})/6 = (4 + 1 + 4 + 0 + 5 + 5)/6 = 19/6 = 5/6 = 30/36 = 30 = 2;$
$C_1 = c(5^{-1*j})/6 = (4*5^{-0*1} + 1*5^{-1*1} + 0*5^{-2*1} + 4*5^{-3*1} + 5*5^{-4*1} + 5*5^{-5*1})/6 = (4 + 3/15 + 24/750 + 20/2500 + 25/15625)/6 = (4 + 3 + 24 + 20 + 25)/6 = 76/6 = 456/36 = 456 = 1;$
$C_2 = c(5^{-2*j})/6 = (4*5^{-0*2} + 1*5^{-1*2} + 0*5^{-2*2} + 4*5^{-3*2} + 5*5^{-4*2} + 5*5^{-5*2})/6 = (4 + 2 + 4 + 10 + 20)/6 = 40/6 = 240/36 = 240 = 2;$
…
c_f(4,1,0,4,5,5) => C_f(2,1,2,1,0,5). Выделенные символы были бы нулями, если бы ошибки не было. Теперь видно, что ошибка была. В данном случае символы 2,1,0,5 называются синдромом ошибки.

Алгоритм Берлекампа-Месси для вычисления позиции ошибки

Чтобы исправить ошибку, нужно знать, какие именно символы были переданы с ошибкой. На этом этапе вычисляется, где находятся ошибочные символы, сколько было ошибок, а также можно ли исправить такое количество ошибок.
Алгоритм Берлекампа-Месси занимается поиском полинома, который, при перемножении на специальную матрицу, подготовленную из чисел синдрома (пример далее), отдаст нулевой вектор. Доказательство алгоритма показывает, что корни

этого полинома содержат информацию о позиции символов с ошибками в полученном кодовом слове.
Так как максимальное количество ошибок для рассматриваемого случая может быть 2, напишем формулу искомого полинома в матричном виде для двух ошибок (полином степени 2): Г = [1 Г1 Г2].
Теперь запишем синдром (2,1,0,5) в формате матрицы Тёплица. Если вы посмотрите на синдром и на полученную матрицу, вы сразу заметите принцип создания такой матрицы. Размерность матрицы обусловлена полиномом Г, обозначенным выше.

$$\begin{bmatrix} 0 & 5 & ? \\ 1 & 0 & 5 \\ 2 & 1 & 0 \end{bmatrix}$$

Уравнение, которое нужно решить:

$$[1 \quad Г1 \quad Г2]\begin{bmatrix} 0 & 5 & ? \\ 1 & 0 & 5 \\ 2 & 1 & 0 \end{bmatrix} = [0 \quad 0 \quad ?]$$

Элементы под знаками вопроса не влияют на результат. Необходимо найти Г1 и Г2, они отвечают за позиции ошибок. Мы постепенно будем увеличивать размерность, с которой работаем. Начинаем с 1 и с нижнего левого края матрицы (отмечено зеленым) при минимальной размерности 1x1.

$$[1] * [2] = [2]$$

Увеличиваем размерность. Предположим, что ошибки в позиции Г1 у нас нет. Тогда Г1 = 0.

$$[1 \quad 0]\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = [1 \quad 0]$$

Мы должны получить [0 0] справа, или, как минимум, один 0 на первом месте, чтобы повысить размерность вычислений, однако на первом месте стоит 1. Мы можем подобрать такое значение для Г1 (которое сейчас 0), чтобы справа получить необходимый 0. Процесс подбора значения Г1 в алгоритме оптимизирован. Запишем два уравнения, рассмотренных ваше, еще раз, а также выведем третье уравнение, которым вычисляется Г1. Цветами показано, что куда идет.

$$[1] * [2] = [2]; \qquad [1 \quad 0]\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = [1 \quad 0] \qquad => \qquad [1 \quad 0] - \frac{1}{2}[0 \quad 1] = [1 \quad 3]$$

Т.е. Г1 = 3. Напоминаю, что подсчет идет в GF(7). Также стоит отметить, что значение Г1 временное. Во время расчетов Г2 оно может поменяться. Считаем заново правую часть:

$$[1 \quad 3]\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = [0 \quad 3]$$

Получили 0 справа, теперь повышаем размерность. Предполагаем по аналогии, что Г2 = 0. Используем найденный Г1=3.

$$[1 \quad 3 \quad 0]\begin{bmatrix} 0 & 5 & ? \\ 1 & 0 & 5 \\ 2 & 1 & 0 \end{bmatrix} = [3 \quad 5 \quad ?]$$

Первым делом вместо тройки справа нужно получить 0. Действия соответствуют предыдущим. Далее подбор значения Г2.

$$[1] * [2] = [2] \qquad [1 \quad 3 \quad 0]\begin{bmatrix} 0 & 5 & ? \\ 1 & 0 & 5 \\ 2 & 1 & 0 \end{bmatrix} = [3 \quad 5 \quad ?]$$

$$[1 \quad 3 \quad 0] - \frac{3}{2}[0 \quad 0 \quad 1] = [1 \quad 3 \quad 2]$$

Запишем новое значение Г2=2 в основное уравнение и опять попробуем найти значения справа:

$$[1 \quad 3 \quad 2]\begin{bmatrix} 0 & 5 & ? \\ 1 & 0 & 5 \\ 2 & 1 & 0 \end{bmatrix} = [0 \quad 0 \quad ?]$$

Задача на данном шаге была получить только первый ноль, однако волей случая второй элемент матрицы также оказался нулем, т.е. задача решена. Если бы он не был нулем, мы бы еще раз делали подбор значений (на этот раз и для Г1, и для Г2), повысив размерность подбора. Если вы заинтересовались данным алгоритмом, вот еще два примера.
Итак, Г1 = 3, Г2 = 2. Ненулевые значения для Г1 и Г2 показывает, что было 2 ошибки. Запишем матрицу Г в виде полинома: $Г(z) = 1 + 3x + 2x^2$. Корни с учетом степеней примитивного элемента, в которых результат получается 0:
$Г(5^3) = 1 + 3*6 + 2*6^2 = 91 = 13*7 = 0$.
$Г(5^5) = 1 + 3*3 + 2*3^2 = 28 = 0$.
Это означает, что ошибки в позициях 3 и 5.
Аналогично можно найти остальные значения, чтобы убедиться, что они не дадут нули:
$Г=>г(z^j) = (3,5,5,0,4,0)$. Как вы могли заметить, снова используется IDFT в GF(7).

Исправление ошибок методом Форни

На предыдущем этапе вычислялись позиции ошибок, теперь осталось найти верные значения. Полином, описывающий позиции ошибок: $Г(z) = 1 + 3x + 2x^2$. Запишем его в нормализированном виде, умножив на 4 и воспользовавшись свойствами GF(7): $Г(z) = x^2 + 5x + 4$.
Метод Форни основан на интерполяции Лагранжа и использует полиномы, которыми мы оперировали в алгоритме Берлекампа-Месси.
В методе дополнительно рассчитываются те символы, которые стоят на местах, не относящихся к синдрому. Это те позиции, которые соответствуют реальным значениям, однако для них рассчитываются другие значения, которые получаются из свертки синдрома ошибки и полинома Г. Эти вычисленные новые значения вместе с синдромом формируют маску ошибки. Далее

выполняется IDFT и результат представляет собой непосредственно ошибку, которая ранее суммировалась с передаваемым кодовым словом. Она вычитается из полученного слова и мы получаем первоначальное переданное слово. Затем выполняем DFT для переданного кодового слова и получаем, наконец, информацию. Далее — как это происходит в контексте рассматриваемого примера.

Запишем вектор ошибки F (последние 4 символа — синдром, который мы постоянно используем, два знака вопроса — на местах информационных символов, это маска ошибки) и обозначим каждый символ буквой:

?	?	2	1	0	5
F0	F1	F2	F3	F4	F5

Символы полинома локатора ошибки Г(z) = x2 + 5x + 4 обозначим так:

4	5	1
Г0	Г1	Г2

Перемножение полинома Г на матрицу Тёплица в предыдущем параграфе было, по сути, операцией циклической свертки: если расписать линейные уравнения, которые получаются из матричного, можно увидеть, что значения, которые берутся из синдрома (значения матрицы Тёплица), от уравнения к уравнению просто меняются местами, двигаясь последовательно в определенную сторону, а это и называется сверткой. Я специально разместил полиномы F и Г друг над другом в начале этого абзаца, чтобы можно было делать свертки (перемножать поэлементно в определенном порядке), двигая полиномы визуально. Раскрывая матричное уравнение из предыдущего параграфа и используя обозначения для полиномов F и Г, введенные только что:
Г0*F4 + Г1*F3 + Г2*F2 = 0
Г0*F5 + Г1*F4 + Г2*F3 = 0
Ранее свертка производилась только для синдрома, в методе Форни нужно сделать свертки для F0 и F1, а после найти их значения:
Г0*F3 + Г1*F2 + Г2*F1 = 0
Г0*F2 + Г1*F1 + Г2*F0 = 0
F0 = -Г0*F3 – Г1*F2 = 0
F1 = -Г0*F2 – Г1*F1 = 6
То есть F = (6,0,2,1,0,5). Проводим IDFT, так как ошибка суммировалась со словом, которое было в кодированном IDFT виде: f = (0,0,0,2,0,6).
Вычитаем ошибку f из полученного кодового слова cf: (4,1,0,4,5,5) — (0,0,0,2,0,6) = c(4,1,0,2,5,6)
Сделаем DFT для этого слова: c(4,1,0,2,5,6) => C=(3,1,0,0,0,0). А вот и наши символы 3 и 1, которые нужно было передать.

Заключение

Обычно используются расширенные коды Рида-Соломона, то есть поле Галуа представляет собой степень двойки ($GF(2^m)$), например кодирование байтов информации. Алгоритмы работы похожи на те, что разобраны в этой статье.
Есть множество разновидностей алгоритма в зависимости от областей применения, а также в зависимости от возраста каждой конкретной разновидности и от компании-разработчика. Чем алгоритм моложе, тем он сложнее.

Также много устройств используют готовые таблицы ошибок, рассчитанные заранее. В условиях использования арифметики Галуа получается конечное количество возможных ошибок. Это свойство и используется для уменьшения количества расчетов. Здесь в случае, если синдром получается ненулевой, он просто сравнивается с таблицей возможных ошибочных синдромов.

Курс теории кодирования для многих часто является одним из самых сложных. Буду рад, если данная статья кому-нибудь поможет разобраться в данной теме быстрее.

Теги: теория кодирования, коды Рида-Соломона

Комментарии 32

Спасибо за статью! К сожалению, в интернете очень сложно найти понятные материалы про коды Рида-Соломона на русском языке. Ваша же статья понравилась выделением цветами «переносящихся» чисел, ссылками, а также достаточно понятным изложением с помощью простого примера.
Кстати стоит добавить, что также, именно благодаря кодам Рида-Соломона, читаются большинство матричных штрихкодов (такие, как QR Code, Datamatrix и прочие). А также позволяет делать внутри них рисунки, сохраняя возможность правильного распознавания.

yTko 26 августа 2013 в 19:25 0

Прочитал сначала как «Коты Рида-Соломона» из-за картинки :)
С кодами РС знаком, почти весь семестр занимался ими + летняя практика.
Возможно лекции были очень хорошие и я все относительно быстро освоил. Но эта статья меня только испугала наличием абсолютно незнакомых названий и методов (вполне возможно, что все они носят несколько названий или применяются в разных задачах (теория и практика => разные подходы). Например, процедура Ченя у нас была для поиска позиции ошибки), поэтому я и не знаю как реагировать на статью.
Но все равно спасибо, русскоязычной и наглядной информации действительно мало в интернете.

alcanoid 26 августа 2013 в 20:59 0

Коты, однозначно! Картинка провокационная. :)

akuzmin 26 августа 2013 в 22:07 0

Там в alt'е изображения даже написано, кому котэ принадлежит :)

> Например, процедура Ченя у нас была для поиска позиции ошибки

Методов сейчас много, мы изучали (скорее всего, вы тоже) по два-три алгоритма для каждого этапа, я выбрал те, которые объяснялись наиболее подробно. Кстати, алгоритм Берлекампа-Месси — это первый открытый алгоритм поиска позиции ошибки, после его открытия коды Рида-Соломона получили возможность применяться на практике.

yTko 26 августа 2013 в 22:30 0

Да вроде нет, нам может быть про них говорили, а в итоге рассказывали что-то общее, ну или я про что-то другое пропустил мимо ушей.
Общий алгоритм у нас был примерно такой:
1. Находим порождающим полином g(x) на основе параметра l_0
2. Брали m(x) какое-нибудь — сообщение
3. Затем находили c(x) * x^r mod g(x)
4. Находили кодовое слово a(x) = m(x) * x^r + c(x)
5. Делали в кодовом слове 2 или 1 ошибку, т.е. y(x) = a(x) + e(x)
6. Затем находили синдромы S1-S4
7. Находим через синдромы локаторы ошибок
8. Через локаторы ошибок находим позиции ошибок
9. Затем через манипуляции с матрицами из синдромов находим величины ошибок на позициях из шага 8
10. Складываем коэф из y(x) с величинами ошибок из шага 9 на позициях из шага 8
11. PROFIT!

При условии постоянной практики, подобные задачи решались минут за 30, а иногда и на скорость, чтобы получить халявные баллы в общий зачет по предмету.
Причем вся математика сводилась к сумме степеней элементов из поля галуа или же сумме бинарных векторов + работа с матрицами. Все. Поэтому смотреть на эти вот преобразования фурье мне страшно.

yTko 26 августа 2013 в 22:44 0

глянул в лекции. у нас был Алгоритм Питерсона — Горенстейна — Цирлера (ПГЦ).

akuzmin 26 августа 2013 в 23:58 +1

Мы тоже изучали метод с порождающим полиномом g(x), при этом он у нас выводился из формулы прямого дискретного преобразования Фурье.
Вообще, все коды Рида-Соломона имеют в основе дискретное преобразование Фурье.
Ваш алгоритм похож на тот, который мы изучали.
Методы, которые у нас были по кодированию: 1) чистое DFT; 2) несистематическое годирование с g(x); 3)систематическое кодирование с g(x) и вычислением остатка; 4) систематическое кодирование с полиномом парности.
Методы по нахождению позиции ошибки: 1) метод с помощью построения кривых Prony; 2) Берлекамп-Месси; 3) метод эвклидового деления в двух разновидностях.
Методы по нахождению значений ошибок: 1) рекурсивный метод; 2)метод Форни

yTko 27 августа 2013 в 01:00 +1

сурово, что я еще могу сказать)
хотя, возможно, нам давали более общее понятие всего этого только из-за того, что специальность настолько широка в обхвате, что прям обидно.
А на каких-нибудь более узких специальностях давали более широкое описание всех этих премудростей…

A1ien 26 августа 2013 в 19:39 +1

С Фурье почему немного не так. Бесконечный периодический сигнал раскладывается в ряд Фурье(конечное количество синусоид), а конечный сигнал переходит в частотную область при помощи интеграла Фурье, и представляет из себя не ряд а спектральную плотность, что по сути является бесконечным количеством синусоид.

mayorovp 26 августа 2013 в 19:51 +2

Плюс еще одна поправка — в частотной области функция комплексная, а не вещественная. Иными словами, «зная список частот», мы сигнал не восстановим — нам нужен еще и список фаз.

A1ien 26 августа 2013 в 20:00 0

Дельное замечание:)

akuzmin 26 августа 2013 в 21:06 0

Да, вы правы, однако то предложение служило лишь целью описать примерно. Нет особой нужды использовать преобразование Фурье для периодического сигнала, поэтому по умолчанию я имел в виду непериодический сигнал, который несет какую-то информацию. Что касается бесконечного числа синусоид, следующего из спектральной плотности: это бы лишь усложнило понимание концепции для читающего. По ссылке, приведенной в том же абзаце, автор также объясняет на примере конечного числа синусоид. К тому же, далее у нас рассматривается именно дискрет ное преобразование, что даст на выходе конечный набор значений.
Возможно, есть смысл расписать преобразование Фурье для сигналов более подробно, и вместе с тем несложно, подумаю над этим.

A1ien 26 августа 2013 в 21:20 0

Что касается дискретного преобразование -то его смысл в приближонном представлении спектральной плотности для оптимизации вычислительных ресурсов, в реальности же при применении ДПФ всегда нужно помнить про погрешности.

mayorovp 26 августа 2013 в 21:39 +1

> в реальности же при применении ДПФ всегда нужно помнить про погрешности

Вы ошибаетесь. ДПФ над конечным полем (а автором рассматривается именно оно) не дает никаких погрешностей.

mayorovp 26 августа 2013 в 19:48 +4

> Сложение: 1+2=3; 4+5=2. В общем, результаты всегда хог’ятся.

Простите, но что и с чем здесь поксорилось? Больше похоже на сложение по модулю.

> Вычитание, умножение – так же.

Что — «так же»? Сложение — xor, и умножение — xor? Не может такого быть. Дали бы хоть ссылку на Википедию, если не получается объяснить понятными словами.

Кстати, то, что вы привели — это поле Z_n, известное так же как поле вычетов. А в кодах Рида-Соломона используются более общий вариант полей Галуа — поля полиномов над Z_n.

akuzmin 26 августа 2013 в 21:24 0

> Больше похоже на сложение по модулю.

Вы правы, это сложение по модулю. Исправил, также расписал более подробно арифметические операции.

> А в кодах Рида-Соломона используются более общий вариант полей Галуа — поля полиномов над Zn.

Поле полиномов над Zn используется в расширенных кодах Рида-Соломона, у меня рассмотрены просто коды Рида-Соломона.

nikitadanilov 27 августа 2013 в 02:02 +1

> Количество чисел в поле должно являться простым числом.
Не обязательно. Число элементов конечного поля может быть любой положительной степенью любого простого числа.

akuzmin 27 августа 2013 в 02:17 0

> Не обязательно. Число элементов конечного поля может быть любой положительной степенью любого простого числа.

Вы описываете расширенные коды Рида-Соломона, где оперируют полиномами над конечным полем. Для рассмотренных в статье кодов Рида-Соломона могут быть использованы только поля с размерностями из простых чисел, как было указано. Основное отличие между расширенными и простыми кодами: в простых используется примитивный элемент, в расширенных — примитивный полином.

nikitadanilov 27 августа 2013 в 02:24 +2

Замечание не про RS-коды, а про поля Галуа. У вас написано, что в них число элементов всегда простое, а это не так.

akuzmin 27 августа 2013 в 02:52 0

Спасибо, исправил в статье.

qw1 20 сентября 2013 в 17:19 0

Я думал, что поле — это тогда и только тогда, если кол-во элементов — простое число.
Например 4=2^2, множество { 0, 1, 2, 3 } не поле, потому что 2*2=0, как и 2*0=0 и операция деления неоднозначна (0/2=0, 0/2=2).

nikitadanilov 20 сентября 2013 в 17:30 +1

В полях с p^q элементами, q > 1, умножение это не «обычное» умножение по модулю p, оно устроено несколько сложнее.

qw1 20 сентября 2013 в 18:11 0

А, почитал вики. Для умножения 2*2 кодируем { 0, 1, 2, 3 } через { 0, 1, x, x+1 },
тогда x*x=x+1 (по модулю x+1), декодируем обратно в 3.

Если такие трюки разрешены, зачем останавливаться на многочленах одной переменной.
Кто знает, может, кодируя многочленами от x и y, у нас в поле будет не p^k, а другое число элементов.

mayorovp 20 сентября 2013 в 18:27 +1

Я знаю — все равно будет p^k. Азы комбинаторики… Кроме того, есть еще и теорема, говорящая о том, что кроме как p^k элементов в конечном поле получить просто невозможно.

qw1 20 сентября 2013 в 18:42 0

Считаем… Поле Z2, макс. степень 2
0, 1, x, y, x+1, y+1, xy, xy+1
8 элементов. Что-ж, контрпример не удался. Придётся согласиться))

qw1 20 сентября 2013 в 18:58 0

И правда, коэф. при любом слагаемом пробегает от 0 до p-1, всего k слагаемых.
Итого, p^k многочленов.
Пример для двух переменных с коэф. из Z3

xy	y	x	1	
0	0	0	0	0
0	0	0	1	1
0	0	0	2	2
0	0	1	0	x
...				
1	2	0	2	xy+2y+2
...				
2	2	2	2	2xy+2x+2y+2

nikitadanilov 20 сентября 2013 в 20:09 +1

Доказательство очень простое. Пусть поле F состоит из N элементов. Его характеристика (https://en.wikipedia.org/wiki/Characteristic_(algebra)) не может быть 0, т.к. поле конечно. Значит характеристика это простое число p. Значит F содержит подполе Z/pZ из p элементов. Рассмотрим F как векторное пространство над полем скаляров Z/pZ. Его размерность конечна, т.к. F конечно. Пусть размерность равна n. Тогда всякий элемент F однозначно представляется координатами (f1, f2, ..., fn), где fi принадлежит Z/pZ. Значит в F всего имеется ровно p^n элементов.

qw1 20 сентября 2013 в 20:58 0

Изящно. Про характеристику не знал.

mayorovp 20 сентября 2013 в 17:34 +1

Да, кольцо вычетов Z_p является полем только при простом p. Но не забывайте, что существуют еще и кольца полиномов $Z_p[x]/f_k(x)$, которые также являются полями при простом p и неприводимом $f_k(x)$. Число элементов такого поля как раз и есть p^k

Veter9 27 августа 2013 в 09:42 +2

Весьма круто(и как обычно местами непонятно) и интересно написано. Пиши еще, мне понравился стиль изложения.

Aux 27 августа 2013 в 10:26 –5

Это всё хорошо, но царапины на дисках ни черта не читаются…

DIHALT 28 августа 2013 в 12:37 0

Еще у Криса Касперского была хорошая статья про это.