Летняя суперкомпьютерная академия – 2012

Лукция 3. Метод Монтгомери vs. метод Видемана-Копперсмита

Замарашкин Николай Леонидович

Институт вычислительной математики РАН

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита

В поиске параллельного алгоритма

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита

Задана матрица $H \in \mathbb{F}_2^{m \times n}$, с $m < n$.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита

Задана матрица $H \in \mathbb{F}_2^{m \times n}$, с $m < n$. Найти нетривиальный вектор $c \in \mathbb{F}_2^n$:

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита

Задана матрица $H \in \mathbb{F}_2^{m \times n}$, с $m < n$. Найти нетривиальный вектор $c \in \mathbb{F}_2^n$:

$$Hc = 0. \tag{1}$$

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита

Задана матрица $H \in \mathbb{F}_2^{m \times n}$, с $m < n$. Найти нетривиальный вектор $c \in \mathbb{F}_2^n$:

$$Hc = 0. \tag{1}$$

Считается, что матрица H является большой разреженной матрицей.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Что заставляет нас искать методы, отличные от метода Монтгомери?

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Что заставляет нас искать методы, отличные от метода Монтгомери?

- необходимость наличия высокоскоростной сети обмена данными между вычислительными узлами системы (обмен данными имеет порядок $\mathcal{O}(N^2/\sqrt{\text{число процессоров}})$)

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Что заставляет нас искать методы, отличные от метода Монтгомери?

- необходимость наличия высокоскоростной сети обмена данными между вычислительными узлами системы (обмен данными имеет порядок $\mathcal{O}(N^2/\sqrt{\text{число процессоров}})$)
- непараллельная "сущность"алгоритма Ланцоша (построение A-ортогонального базиса пространства Крылова последовательная процедура)

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Вывод: необходимо уйти от явной процедуры вычисления A-ортогонального базиса?

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Вывод: необходимо уйти от явной процедуры вычисления A-ортогонального базиса?

Возможно ли это? Существует ли метод, обладающий лучшими параллельными характеристиками?

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Вывод: необходимо уйти от явной процедуры вычисления A-ортогонального базиса?

Возможно ли это? Существует ли метод, обладающий лучшими параллельными характеристиками?

Ответ положительный! Один такой метод хорошо известен и связан с именами Видемана и Копперсмита.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсимта

Вывод: необходимо уйти от явной процедуры вычисления A-ортогонального базиса?

Возможно ли это? Существует ли метод, обладающий лучшими параллельными характеристиками?

Ответ положительный! Один такой метод хорошо известен и связан с именами Видемана и Копперсмита.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана.

Пусть

$$Ax = b, \quad A \in \mathbb{F}_2^{N \times N}$$

согласованная система линейных уравнений над $\mathbb{F}_2$ с квадратной невырожденной матрицей A.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана.

Пусть

$$Ax = b, \quad A \in \mathbb{F}_2^{N \times N}$$

согласованная система линейных уравнений над $\mathbb{F}_2$ с квадратной невырожденной матрицей A.

Предположим, что нам удалось найти характеристический (минимальный) полином

$$f(\lambda) = 1 + \sum_{i=1}^{N_f} f_i \lambda^i.$$

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана.

Пусть

$$Ax = b, \quad A \in \mathbb{F}_2^{N \times N}$$

согласованная система линейных уравнений над $\mathbb{F}_2$ с квадратной невырожденной матрицей A.

Предположим, что нам удалось найти характеристический (минимальный) полином

$$f(\lambda) = 1 + \sum_{i=1}^{N_f} f_i \lambda^i.$$

По теореме Кэли-Гамильтона

$$f(A) = 0 \rightarrow f(A)b = 0 \rightarrow b + \sum_{i=1}^{N_f} f_i A^i b = 0$$

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

Таким образом,

$$b = \sum_{i=1}^{N_f} f_i A^i b = A \left(\sum_{i=0}^{N_f - 1} f_{i+1} A^i b \right)$$

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

Таким образом,

$$b = \sum_{i=1}^{N_f} f_i A^i b = A \left(\sum_{i=0}^{N_f - 1} f_{i+1} A^i b \right)$$

и можно выбрать x в виде

$$x = \sum_{i=0}^{N_f - 1} f_{i+1} A^i b,$$

и $x \in \mathcal{L} = \text{Span}\{b, Ab, A^2 b, \cdots, A^{N_f - 1} b\}$.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

Как искать $f(\lambda)$?

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

Как искать $f(\lambda)$?

Выберем случайный вектор y и заметим, что

$$y^T \left(\sum_{i=0}^{N_f} f_i A^i \right) A^j b = 0, \quad j = 0, \cdots, N-1 \qquad (2)$$

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

Как искать $f(\lambda)$?

Выберем случайный вектор y и заметим, что

$$y^T \left(\sum_{i=0}^{N_f} f_i A^i \right) A^j b = 0, \quad j = 0, \cdots, N-1 \tag{2}$$

Из (2) следует, что f_i удовлетворяют системе линейных уравнений

$$\sum_{i=0}^{N_f} f_i \alpha_{i+j} = 0, \quad j = 0, \cdots, N-1, \tag{3}$$

где $\alpha_i = y^T A^i b \in \mathbb{F}_2$, $i = 0, \cdots, 2 \cdot N + N_f - 1$.

Монтгомери vs. Видеман-Копперсмит

2. Метод Видемана-Копперсмита. Идея Видемана

$$\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{N_f-1} \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{N_f} \\ \alpha_2 & \alpha_3 & \alpha_4 & \cdots & \alpha_{N_f+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{N-1} & \alpha_N & \alpha_{N+1} & \cdots & \alpha_{N+N_f-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \cdots \\ f_{N_f-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}.$$

Решение такой системы специального вида (ганкелева матрица) может быть получено с помощью супербыстрых алгоритмов за время $\mathcal{O}(N \log^{\alpha} N)$.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Пока, однако, не видно, а где же здесь дополнительная параллельность?

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Пока, однако, не видно, а где же здесь дополнительная параллельность?

Необходимо опять ввести блочность!

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Заменим вектор b на набор блоков $B = [B_1 B_2 \cdots B_s]$.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Заменим вектор b на набор блоков $B = [B_1 B_2 \cdots B_s]$.

Какие следует взять размеры блоков?

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Заменим вектор b на набор блоков $B = [B_1 B_2 \cdots B_s]$.

Какие следует взять размеры блоков?

Аналогично вектор y заменим на блок Y.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Заменим вектор b на набор блоков $B = [B_1 B_2 \cdots B_s]$.

Какие следует взять размеры блоков?

Аналогично вектор y заменим на блок Y.

Получим

$$Y^T A^j \left(\sum_{i=0}^{N_f} A^i B f_i \right) = 0, \quad j = 0, \cdots, \Delta_2 \tag{4}$$

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Заменим вектор b на набор блоков $B = [B_1 B_2 \cdots B_s]$.

Какие следует взять размеры блоков?

Аналогично вектор y заменим на блок Y.

Получим

$$Y^T A^j \left(\sum_{i=0}^{N_f} A^i B f_i \right) = 0, \quad j = 0, \cdots, \Delta_2 \tag{4}$$

Из (4) следует, что f_i, которые теперь блоки, удовлетворяют системе линейных уравнений

$$\sum_{i=0}^{N_f} \alpha_{i+j} f_i = 0, \quad j = 0, \cdots, \Delta_2, \tag{5}$$

где $\alpha_i = Y^T A^i B \in \mathbb{F}_2^{n_y \times n_b}$, $i = 0, \cdots, 2 \cdot \Delta_1 + \Delta_2$.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Уравнение (5) эквивалентно системе уравнений

$$\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{\Delta_1-1} \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{\Delta_1} \\ \alpha_2 & \alpha_3 & \alpha_4 & \cdots & \alpha_{\Delta_1+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{\Delta_2-1} & \alpha_{\Delta_2} & \alpha_{\Delta_2+1} & \cdots & \alpha_{\Delta_1+\Delta_2-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \cdots \\ f_{\Delta_1-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

со специальной матрицей. Стоимость решения такого уравнения, благодаря хитрым методам, можно сделать почти пропорциональной размеру системы N.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$);

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельный алгоритм
2. решить блочную ганкелеву систему

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельный алгоритм
2. решить блочную ганкелеву систему (сложность $\mathcal{O}(N \log^{\alpha} N)$;

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельный алгоритм
2. решить блочную ганкелеву систему (сложность $\mathcal{O}(N \log^{\alpha} N)$; не требуется параллельности
3. получить решение первоначальной системы

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельный алгоритм
2. решить блочную ганкелеву систему (сложность $\mathcal{O}(N \log^{\alpha} N)$; не требуется параллельности
3. получить решение первоначальной системы (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$);

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Метод Видемана-Копперсмита:

1. составить блочную ганкелеву систему (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельный алгоритм
2. решить блочную ганкелеву систему (сложность $\mathcal{O}(N \log^{\alpha} N)$; не требуется параллельности
3. получить решение первоначальной системы (сложность $\mathcal{O}(\text{число ненулей в строке} \cdot N^2)$); требуется параллельность

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Составить блочную ганкелеву матрицу:

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Составить блочную ганкелеву матрицу:

- абсолютно параллельная процедура

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Составить блочную ганкелеву матрицу:

- абсолютно параллельная процедура
- количество пересылок $=$ собрать матрицу ганакеля на одном узле $\Rightarrow$

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Составить блочную ганкелеву матрицу:

- абсолютно параллельная процедура
- количество пересылок $=$ собрать матрицу ганакеля на одном узле $\Rightarrow \mathcal{O}(N)$, а не $O(N^2)$!
- идеально подходит для GRID систем

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Решить блочную ганкелеву систему:

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Решить блочную ганкелеву систему:

1. Это, безусловно, наиболее наукоемкая часть задачи, но ее сложность низка, и она не имеет, вообще говоря, отношения к собственно параллельным вычислениям

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Формула для получения такого решения имеет вид:

$$X = \sum_{i=0}^{\Delta_1} A^i B f_{i-1}.$$

И что тут страшного, спросите вы?

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Формула для получения такого решения имеет вид:

$$X = \sum_{i=0}^{\Delta_1} A^i B f_{i-1}.$$

И что тут страшного, спросите вы?

- надо строить пространства Крылова

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Формула для получения такого решения имеет вид:

$$X = \sum_{i=0}^{\Delta_1} A^i B f_{i-1}.$$

И что тут страшного, спросите вы?

- надо строить пространства Крылова (вспомните, как мы их строили в первой части алгоритма)

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Формула для получения такого решения имеет вид:

$$X = \sum_{i=0}^{\Delta_1} A^i B f_{i-1}.$$

И что тут страшного, спросите вы?

- надо строить пространства Крылова (вспомните, как мы их строили в первой части алгоритма)
- вообще говоря, требуются обмены для векторов Крылова

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Получить решение первоначальной системы.

Формула для получения такого решения имеет вид:

$$X = \sum_{i=0}^{\Delta_1} A^i B f_{i-1}.$$

И что тут страшного, спросите вы?

- надо строить пространства Крылова (вспомните, как мы их строили в первой части алгоритма)
- вообще говоря, требуются обмены для векторов Крылова

Так что же, метод не так уж и хорош?

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

На помощь приходит out-of-core параллелизм.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

На помощь приходит out-of-core параллелизм.

Подсчитаем память для крыловских пространств:

$$\text{память} = 10^{16}/64 \text{ слов.}$$

Построим паралельный метод.

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

1. параллельный метод

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

1. параллельный метод
2. допускает реализацию на GRID системах

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

1. параллельный метод
2. допускает реализацию на GRID системах

Недостатки метода Видемана-Копперсмита:

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

1. параллельный метод
2. допускает реализацию на GRID системах

Недостатки метода Видемана-Копперсмита:

1. высокая сложность эффективной реализации

Монтгомери vs. Видеман-Копперсмит

1. Метод Видемана-Копперсмита. Идея Копперсмита.

Достоинства метода Видемана-Копперсмита:

1. параллельный метод
2. допускает реализацию на GRID системах

Недостатки метода Видемана-Копперсмита:

1. высокая сложность эффективной реализации
2. значительные требования к вычислительной системе на втором шаге

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Какие еще методы используют для решения систем уравнений над $\mathbb{F}_2$?

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Какие еще методы используют для решения систем уравнений над $\mathbb{F}_2$?

Имеется по крайней мере еще одна область знаний, где задача решения систем над $\mathbb{F}_2$ присутствует повсеместно.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Какие еще методы используют для решения систем уравнений над $\mathbb{F}_2$?

Имеется по крайней мере еще одна область знаний, где задача решения систем над $\mathbb{F}_2$ присутствует повсеместно.

Коды, исправляющие ошибки, в задаче передачи информации по каналу с шумом.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Какие еще методы используют для решения систем уравнений над $\mathbb{F}_2$?

Имеется по крайней мере еще одна область знаний, где задача решения систем над $\mathbb{F}_2$ присутствует повсеместно.

Коды, исправляющие ошибки, в задаче передачи информации по каналу с шумом.
Нас особенно будут интересовать методы решения, используемые в этой области.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Что такое задача передачи информации по каналу с шумом.

$$\cdots 01101001000 \cdots \rightarrow \text{ CH } \rightarrow \cdots 01001001110 \cdots$$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Что такое задача передачи информации по каналу с шумом.

$$\cdots 01101001000\cdots \rightarrow \text{ CH } \rightarrow \cdots 01001001110\cdots$$

Как передавать информацию без искажений?

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Что такое задача передачи информации по каналу с шумом.

$$\cdots 01101001000\cdots \rightarrow \text{ CH } \rightarrow \cdots 01001001110\cdots$$

Как передавать информацию без искажений?

Когда это возможно?

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Пусть $\mathcal{X}$, $\mathcal{Y}$ — два конечных множества.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Пусть $\mathcal{X}$, $\mathcal{Y}$ — два конечных множества.

Будем называть:

- $\mathcal{X}$ – входным алфавитом

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Пусть $\mathcal{X}$, $\mathcal{Y}$ — два конечных множества.

Будем называть:

- $\mathcal{X}$ – входным алфавитом
- $\mathcal{Y}$ – выходным алфавитом

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Пусть $\mathcal{X}$, $\mathcal{Y}$ — два конечных множества.

Будем называть:

- $\mathcal{X}$ – входным алфавитом
- $\mathcal{Y}$ – выходным алфавитом

Элементы алфавитов называются символами.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Клод Шеннон – человек, который создал математическую теорию связи (1946).

Определение. Математический канал связи.

Пусть $\mathcal{X}$, $\mathcal{Y}$ — два конечных множества.

Будем называть:

- $\mathcal{X}$ – входным алфавитом
- $\mathcal{Y}$ – выходным алфавитом

Элементы алфавитов называются символами.

Математический канал связи задается функцией перехода $\mathcal{P}(y|x)$, то есть условной вероятностью принять символ $y \in \mathcal{Y}$, если был отправлен символ $x \in \mathcal{X}$.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Примеры каналов.

1. Канал BSC:
   - 1.1 $\mathcal{X} = \mathcal{Y} = \{0, 1\}$
   - 1.2 $P(0|0) = P(1|1) = 1 - p$, $P(1|0) = P(0|1) = p$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Примеры каналов.

1. Канал BSC:
   - 1.1 $\mathcal{X} = \mathcal{Y} = \{0, 1\}$
   - 1.2 $P(0|0) = P(1|1) = 1 - p$, $P(1|0) = P(0|1) = p$
2. Канал BEC:
   - 2.1 $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1, \delta\}$
   - 2.2 $P(0|0) = P(1|1) = 1 - p$, $P(\delta|0) = P(\delta|1) = p$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Примеры каналов.

1. Канал BSC:
   - 1.1 $\mathcal{X} = \mathcal{Y} = \{0, 1\}$
   - 1.2 $P(0|0) = P(1|1) = 1 - p$, $P(1|0) = P(0|1) = p$
2. Канал BEC:
   - 2.1 $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1, \delta\}$
   - 2.2 $P(0|0) = P(1|1) = 1 - p$, $P(\delta|0) = P(\delta|1) = p$

В дальнейшем мы ограничимся только BSC каналом.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как передавать без ошибок? $\Rightarrow$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как передавать без ошибок? $\Rightarrow$ Избыточность.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как передавать без ошибок? $\Rightarrow$ Избыточность.

Пример (код с повторениями).

$$\cdots 011 \cdots \to \cdots 000\ 111\ 000 \cdots \to \text{ СН} \to \cdots 001\ 111\ 001 \cdots$$

Как оптимизировать избыточность?

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$
2. инъективное отображение $\mathcal{E} : \mathbb{F}_2^k \to \mathcal{C}$, называемое кодером

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$
2. инъективное отображение $\mathcal{E} : \mathbb{F}_2^k \to \mathcal{C}$, называемое кодером
3. сюръективное отображение $\mathcal{D} : \mathbb{F}_2^n \to \mathcal{C}$, называемое декодером

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$
2. инъективное отображение $\mathcal{E} : \mathbb{F}_2^k \to \mathcal{C}$, называемое кодером
3. сюръективное отображение $\mathcal{D} : \mathbb{F}_2^n \to \mathcal{C}$, называемое декодером

будем называть двоичным кодом.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$
2. инъективное отображение $\mathcal{E} : \mathbb{F}_2^k \to \mathcal{C}$, называемое кодером
3. сюръективное отображение $\mathcal{D} : \mathbb{F}_2^n \to \mathcal{C}$, называемое декодером

будем называть двоичным кодом.
Если дополнительно для любых x и y в $\mathcal{C}$ вектор $x + y \in \mathcal{C}$ (сложение в поле $\mathbb{F}_2$),

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Бинарный код.

Рассмотрим множество $\mathbb{F}_2^n$. Произвольное подмножество $\mathcal{C}$ такое, что:

1. $\#\mathcal{C} = 2^k$
2. инъективное отображение $\mathcal{E} : \mathbb{F}_2^k \to \mathcal{C}$, называемое кодером
3. сюръективное отображение $\mathcal{D} : \mathbb{F}_2^n \to \mathcal{C}$, называемое декодером

будем называть двоичным кодом.
Если дополнительно для любых x и y в $\mathcal{C}$ вектор $x + y \in \mathcal{C}$ (сложение в поле $\mathbb{F}_2$), то такой код будем называть линейным двоичным кодом.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Мы далее будем изучать только линейные двоичные коды!

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Пример (код с повторениями).

- $n = 3$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Пример (код с повторениями).

- $n = 3$
- $k = 1$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Пример (код с повторениями).

- $n = 3$
- $k = 1$
- $\mathcal{E}$: $0 \to 000$, $1 \to 111$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Пример (код с повторениями).

- $n = 3$
- $k = 1$
- $\mathcal{E}$: $0 \to 000$, $1 \to 111$
- $\mathcal{D}$: мажоритарный декодер

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Скорость кода.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Скорость кода.

Число R

$$R = \frac{k}{n} \tag{6}$$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Скорость кода.

Число R

$$R = \frac{k}{n} \tag{6}$$

называется скоростью бинарного кода.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Определение. Скорость кода.

Число R

$$R = \frac{k}{n} \tag{6}$$

называется скоростью бинарного кода.

Скорость бинарного кода задает избыточность кода!

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки. Теорема Шеннона.

Теорема Шеннона для BSC канала.

Определим число C

$$C = 1 - p \log_2 p - (1-p) \log_2 (1-p),$$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки. Теорема Шеннона.

Теорема Шеннона для BSC канала.

Определим число C

$$C = 1 - p \log_2 p - (1-p) \log_2 (1-p),$$

называемое пропускной способностью канала. Для любого ε существует линейный бинарный код:

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки. Теорема Шеннона.

Теорема Шеннона для BSC канала.

Определим число C

$$C = 1 - p \log_2 p - (1-p) \log_2 (1-p),$$

называемое пропускной способностью канала. Для любого ε существует линейный бинарный код:

- $R > C - \varepsilon$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки. Теорема Шеннона.

Теорема Шеннона для BSC канала.

Определим число C

$$C = 1 - p \log_2 p - (1-p) \log_2 (1-p),$$

называемое пропускной способностью канала. Для любого ε существует линейный бинарный код:

- $R > C - \varepsilon$
- $\mathcal{P}_b < \varepsilon$

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки. Теорема Шеннона.

Теорема Шеннона для BSC канала.

Определим число C

$$C = 1 - p \log_2 p - (1-p) \log_2 (1-p),$$

называемое пропускной способностью канала. Для любого ε существует линейный бинарный код:

- $R > C - \varepsilon$
- $\mathcal{P}_b < \varepsilon$

Если же $R > C + \varepsilon$, то (несмотря ни на какие усилия) существует δ: $\mathcal{P}_e > \delta$.

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как можно задавать линейный код?

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как можно задавать линейный код? Один из вариантов с помощью проверочной матрицы H:

Линейные системы над $\mathbb{F}_2$

2. Коды, исправляющие ошибки.

Как можно задавать линейный код? Один из вариантов с помощью проверочной матрицы H:

$$Hc = 0. \tag{7}$$