

Задание 1. Алгоритм Loopy Belief Propagation для LDPC-кодов

Курс: Графические модели, весна 2015

Начало выполнения задания: 28 февраля
 Срок сдачи: **15 марта (воскресенье), 23:59.**
 Среда для выполнения задания – MATLAB.

Содержание

Низкоплотностные коды	1
Задача помехоустойчивого кодирования	1
Кодирование с помощью линейного кода	2
Декодирование низкоплотностного кода	2
Оптимизации в процедуре декодирования	3
Формулировка задания	4
Рекомендации по выполнению задания	5
Оформление задания	5

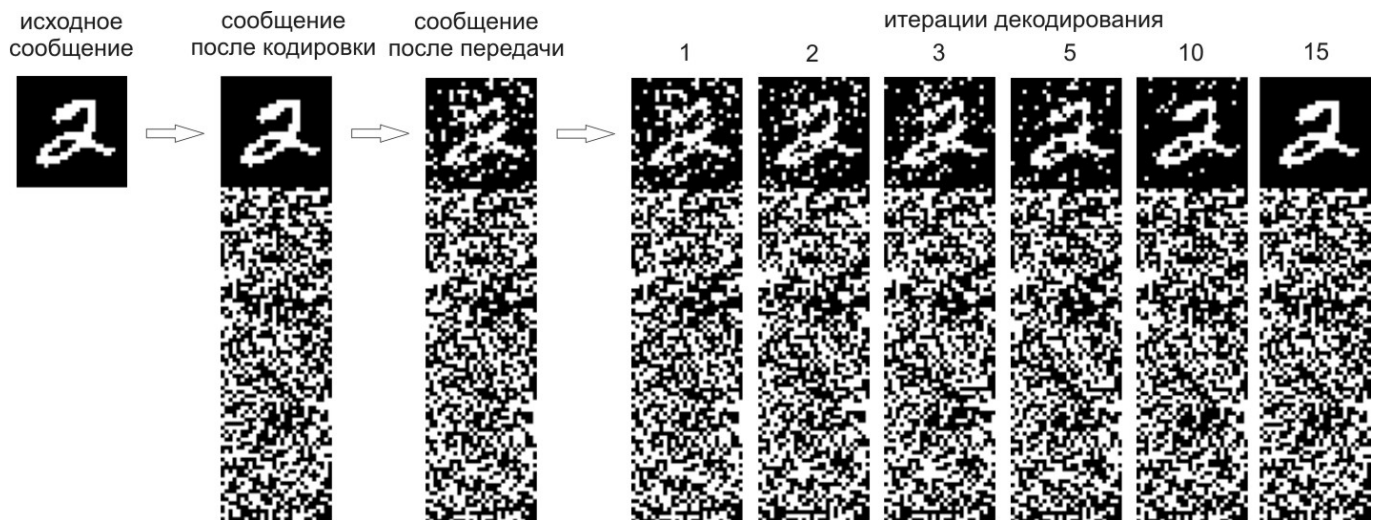


Рис. 1: Демонстрация процесса кодирования/декодирования с использованием низкоплотностного кода. Кодированное сообщение (изображение цифры) содержит значительную избыточность для возможности визуальной оценки процесса декодирования. Алгоритм декодирования не использует свойство избыточности входного сигнала.

Низкоплотностные коды

Задача помехоустойчивого кодирования

Рассмотрим решение задачи безошибочной передачи потока битовой информации по каналу с шумом с помощью кодов, исправляющих ошибки. При *блоковом кодировании* входящий поток информации разбивается на блоки фиксированной длины k , и каждый блок кодируется/декодируется независимо. Обозначим один такой блок через $\mathbf{u} \in \{0, 1\}^k$. Предположим, что во входном потоке данных, вообще говоря, нет избыточности. Поэтому для реализации схемы, способной исправлять ошибки, необходимо закодировать блок \mathbf{u} в некоторое кодовое слово большей длины путем добавления избыточности в передаваемые данные. Обозначим кодовое слово через $\mathbf{v} \in \{0, 1\}^n$, $n > k$. Для кодирования всевозможных блоков \mathbf{u} необходимо использовать 2^k кодовых слов длины n . Назовём (n, k) -*блоковым кодом* C множество 2^k кодовых слов длины n , а величину $r = k/n$ – *скоростью*

кода. При передаче по каналу с шумом кодовое слово \mathbf{v} переходит в принятое слово \mathbf{w} , которое, вообще говоря, отличается от \mathbf{v} . Предположим, что при передаче по каналу длина сообщения не изменяется, т.е. $\mathbf{w} \in \{0, 1\}^n$, а происходит лишь инверсия некоторых бит. Задача алгоритма декодирования состоит в восстановлении по \mathbf{w} переданного слова \mathbf{v} (например, путем поиска среди всевозможных кодовых слов ближайшего к \mathbf{w}). Обозначим результат работы алгоритма декодирования через $\hat{\mathbf{v}}$. На последнем этапе декодированное слово $\hat{\mathbf{v}}$ переводится в декодированное слово исходного сообщения $\hat{\mathbf{u}}$.

Кодирование с помощью (n, k) -линейного блочного кода

Множество $\{0, 1\}^n$ с операциями суммы и произведения по модулю 2 образует линейное пространство над конечным полем из двух элементов $\{0, 1\}$. (n, k) -блочный код C называется *линейным*, если множество его кодовых слов образует линейное подпространство размерности k общего линейного пространства $\{0, 1\}^n$. В этом случае C можно задать с помощью некоторого базиса из k элементов $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k \in \{0, 1\}^n$. Тогда произвольное кодовое слово $\mathbf{v} \in C$ представляется как $\mathbf{v} = \sum_{i=1}^k u_i \mathbf{g}_i$, где $u_i \in \{0, 1\}$ – коэффициенты разложения по базису. Объединим базисные вектора по столбцам в общую матрицу $G = [\mathbf{g}_1, \dots, \mathbf{g}_k] \in \{0, 1\}^{n \times k}$ – *порождающую матрицу кода*. Тогда процедуру кодирования можно представить как $\mathbf{v} = G\mathbf{u}$.

Общее линейное пространство $\{0, 1\}^n$ размерности n раскладывается в прямую сумму линейных подпространств C и C^\perp размерности, соответственно, k и $n - k = m$. Ортогональное подпространство C^\perp можно задать с помощью базиса из m элементов $\mathbf{h}_1, \dots, \mathbf{h}_m \in \{0, 1\}^n$. Тогда условие $\mathbf{v} \in C$ эквивалентно $\mathbf{v}^T \mathbf{h}_j = 0 \forall j = 1, \dots, m$. Объединим базисные вектора \mathbf{h}_j по строкам в общую матрицу

$$H = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \vdots \\ \mathbf{h}_m^T \end{bmatrix} \in \{0, 1\}^{m \times n}.$$

Назовём матрицу H *проверочной матрицей* кода. С её помощью можно проверить, является ли слово \mathbf{v} кодовым словом путём проверки соотношения $H\mathbf{v} = \mathbf{0}$.

Линейный блочный код однозначно задаётся либо порождающей матрицей G , либо проверочной матрицей H . Рассмотрим задачу кодирования слов исходного сообщения \mathbf{u} в кодовые слова \mathbf{v} (n, k) -линейного блочного кода, заданного своей проверочной матрицей H . Над строками матрицы H можно проводить эквивалентные преобразования, соответствующие выбору другого базиса ортогонального подпространства C^\perp . С помощью подобных преобразований матрицу H можно привести к *каноническому ступенчатому виду*. С точностью до перестановки столбцов канонический ступенчатый вид матрицы H эквивалентен её представлению в виде $[I_m \ P]$, где I_m – единичная матрица размера $m \times m$. Тогда в качестве порождающей матрицы кода можно выбрать матрицу

$$G = \begin{bmatrix} P \\ I_k \end{bmatrix}. \quad (1)$$

Действительно, в этом случае $H\mathbf{v} = HG\mathbf{u} = (P + P)\mathbf{u} = \mathbf{0}$. Кодирование по правилу $\mathbf{v} = G\mathbf{u}$ с помощью матрицы (1) будет *систематическим*, т.к. здесь все биты слова \mathbf{u} копируются в фиксированные (последние) биты кодового слова \mathbf{v} . При систематическом кодировании обратный процесс преобразования из декодированного кодового слова $\hat{\mathbf{v}}$ в декодированное сообщение $\hat{\mathbf{u}}$ становится тривиальным.

Декодирование низкоплотного кода

Низкоплотным кодом (или кодом с малой плотностью проверок на чётность) называется бинарный (n, k) -линейный блочный код, в котором проверочная матрица $H \in \{0, 1\}^{m \times n}$ является сильно разреженной.

Рассмотрим бинарный симметричный канал для передачи данных. Здесь при передаче каждый бит независимо инвертируется с некоторой вероятностью q . В результате бинарный симметричный канал задает распределение $p(\mathbf{w}|\mathbf{v})$ для передаваемого кодового слова $\mathbf{v} \in \{0, 1\}^n$ и полученного на выходе слова $\mathbf{w} \in \{0, 1\}^n$ как

$$p(\mathbf{w}|\mathbf{v}) = \prod_{i=1}^n p(w_i|v_i) = \prod_{i=1}^n q^{w_i+v_i} (1-q)^{w_i+v_i+1}.$$

Пропускная способность данного канала определяется величиной $1 + q \log_2 q + (1 - q) \log_2 (1 - q)$.

Объединяя низкоплотный код с бинарным симметричным каналом, получаем следующую вероятностную модель для пары \mathbf{w}, \mathbf{v} :

$$p(\mathbf{v}|\mathbf{w}) \propto p(\mathbf{w}|\mathbf{v}) I[H\mathbf{v} = \mathbf{0}] \propto \prod_{i=1}^n p(w_i|v_i) \prod_{j=1}^m I[\mathbf{h}_j^T \mathbf{v} = \mathbf{0}].$$

Здесь $m = n - k$ – количество проверок на чётность, \mathbf{h}_j^T – j -ая строка матрицы H , а $I[\cdot]$ – индикаторная функция. Фактор-граф введённой модели показан на рис. 2.

Назовём *синдромом* принятого слова \mathbf{w} вектор $\mathbf{s} \in \{0, 1\}^m$, определяемый как $\mathbf{s} = H\mathbf{w}$. Процесс передачи кодового слова по бинарному симметричному каналу можно представить как $\mathbf{w} = \mathbf{v} + \mathbf{e}$, где $\mathbf{e} \in \{0, 1\}^n$ – вектор ошибок ($e_i = 1$, если в позиции i произошла ошибка). Тогда $\mathbf{s} = H\mathbf{w} = H(\mathbf{v} + \mathbf{e}) = H\mathbf{v} + H\mathbf{e} = H\mathbf{e}$. Далее можно перейти от вероятностной модели для переменных \mathbf{v}, \mathbf{w} к аналогичной для переменных \mathbf{e}, \mathbf{s} :

$$p(\mathbf{e}|\mathbf{s}) \propto p(\mathbf{e})p(\mathbf{s}|\mathbf{e}) \propto \prod_{i=1}^n p(e_i) \prod_{j=1}^m I[\mathbf{h}_j^T \mathbf{e} = s_j].$$

Здесь $p(e_i) = q^{e_i}(1-q)^{1-e_i}$. Зная значение вектора ошибок $\hat{\mathbf{e}}$, результат декодирования вычисляется как $\hat{\mathbf{v}} = \mathbf{w} + \hat{\mathbf{e}}$. При использовании вероятностной модели для \mathbf{e}, \mathbf{s} тестирование алгоритма декодирования можно проводить без предварительной реализации процедуры кодирования.

При использовании побитовой функции потерь $\lambda(\mathbf{e}, \hat{\mathbf{e}}) = \sum_{i=1}^n I[e_i \neq \hat{e}_i]$ оптимальная процедура декодирования связана с максимизацией маргиналов отдельных переменных, т.е. $\hat{e}_i = \arg \max p(e_i|\mathbf{s})$. Для поиска маргинальных распределений $p(e_i|\mathbf{s})$ воспользуемся циклическим алгоритмом передачи сообщений (sum-product loop BP) на фактор-графе. Введём обозначения $N(i) = \{j : H_{ji} = 1\}$ – множество факторов, в которых участвует переменная e_i , и $N(j) = \{i : H_{ji} = 1\}$ – множество переменных, которые входят в фактор h_j . Тогда общая схема алгоритма декодирования выглядит следующим образом:

1. Инициализация:

$$\mu_{e_i \rightarrow h_j}(e_i) = p(e_i);$$

2. Пересчёт сообщений от факторов:

$$\mu_{h_j \rightarrow e_i}(e_i) = \sum_{e_k: k \in N(j) \setminus i} I[e_i + \sum_k e_k = s_j] \prod_{k \in N(j) \setminus i} \mu_{e_k \rightarrow h_j}(e_k);$$

3. Пересчёт сообщений от переменных и вычисление beliefs (оценок на маргинальные распределения):

$$\mu_{e_i \rightarrow h_j}(e_i) \propto p(e_i) \prod_{k \in N(i) \setminus j} \mu_{h_k \rightarrow e_i}(e_i);$$

$$b_i(e_i) \propto p(e_i) \prod_{k \in N(i)} \mu_{h_k \rightarrow e_i}(e_i);$$

4. Оценка вектора ошибок:

$$\hat{e}_i = \arg \max b_i(e_i);$$

5. Критерий остановки:

- Если $H\hat{\mathbf{e}} = \mathbf{s}$, то выход алгоритма со статусом 0;
- Если произошла стабилизация по всем b_i , то выход алгоритма со статусом 1;
- Если достигнуто максимальное число итераций, то выход алгоритма со статусом 2;

6. Переход к шагу 2.

Оптимизации в процедуре декодирования

Пересчёт сообщений от факторов

При прямой реализации шага 2 общего алгоритма декодирования требуется рассмотрение для каждого фактора $2^{|N(j)|-1}$ различных конфигураций значений переменных e_k . Это может приводить как к низкой скорости пересчёта сообщений, так и к большим требованиям по памяти.

Рассмотрим более эффективную схему реализации шага 2. Пусть нам необходимо вычислить сообщение $\mu_{h_j \rightarrow e_i}(e_i)$. Перенумеруем все переменные, входящие в j -ый фактор (кроме переменной e_i), как $e_{(1)}, e_{(2)}, \dots, e_{(l)}$, где $l = |N(j)| - 1$. Тогда вычисление сообщения $\mu_{h_j \rightarrow e_i}(e_i)$ можно записать как

$$\mu_{h_j \rightarrow e_i}(e_i) \propto \sum_{e_{(k)}} I[\sum_{k=1}^l e_{(k)} = s_j + e_i] \prod_{k=1}^l \mu_{e_{(k)} \rightarrow h_j}(e_{(k)}).$$

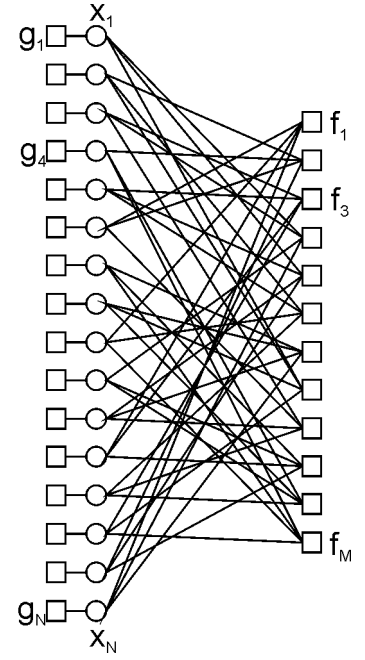


Рис. 2: Фактор-граф для (16, 4)-низкоплотного кода, в проверочной матрице которого в каждом столбце по 3 единицы, а в каждой строке – по 4 единицы.

Данный результат можно интерпретировать как вычисление вероятности $p(\sum_{k=1}^l e_{(k)} = s_j + e_i)$ для набора независимых бинарных переменных $e_{(k)}$ с распределением $p_{(k)}(e_{(k)}) = \mu_{e_{(k)} \rightarrow h_j}(e_{(k)})$. Обозначим через e^k сумму первых k переменных $e_{(k)}$, т.е. $e^k = e_{(1)} + e_{(2)} + \dots + e_{(k)}$. Тогда распределение на e^k можно итерационно пересчитывать по формуле

$$p_k(e^k) = \sum_{e^{k-1}} p_{k-1}(e^{k-1}) p_{(k)}(e^k - e^{k-1}).$$

В результате вычисление $p_l(e^l)$ имеет линейную по l сложность.

Дальнейшее повышение эффективности реализации шага 2 связано с рассмотрением разностей вероятностей $\delta p_{(k)} = p_{(k)}(0) - p_{(k)}(1)$ и $\delta p_k = p_k(0) - p_k(1)$. Нетрудно показать, что

$$\delta p_l = \prod_{k=1}^l \delta p_{(k)}. \quad (2)$$

Зная значение δp_l и учитывая условие нормировки $p_l(0) + p_l(1) = 1$, сами вероятности могут быть вычислены как $p_l(0) = \frac{1}{2}(1 + \delta p_l)$, $p_l(1) = \frac{1}{2}(1 - \delta p_l)$. Таким образом, $\mu_{h_j \rightarrow e_i}(e_i) = p_l(s_j + e_i)$. Основное преимущество условия (2) по сравнению с последовательным пересчётом распределений $p_k(e^k)$ связано с тем, что формула (2) может быть реализована с помощью векторных операций в MATLAB.

Расписание пересчёта сообщений и дэмпфирование

Общая схема циклического алгоритма передачи сообщений оставляет определённый произвол в выборе расписания пересчёта сообщений. Обычно здесь рассматриваются следующие подходы:

- *Параллельное расписание.* В данном случае сначала все вершины посылают сообщения во все факторы, а затем все факторы посылают сообщения во все вершины.
- *Последовательное расписание.* Здесь в цикле по вершинам фактор-графа для данной вершины сначала вычисляются все входящие сообщения от соседних факторов, а затем вычисляются все исходящие сообщения в соседние факторы. Аналогично цикл можно проводить по факторам.

При использовании дэмпфирования с параметром $\lambda \in (0, 1]$ сообщения на итерации t пересчитываются как

$$\begin{aligned} \mu_{h_j \rightarrow e_i}^t(e_i) &= \lambda \mu_{h_j \rightarrow e_i}(e_i) + (1 - \lambda) \mu_{h_j \rightarrow e_i}^{t-1}(e_i); \\ \mu_{e_i \rightarrow h_j}^t(e_i) &= \lambda \mu_{e_i \rightarrow h_j}(e_i) + (1 - \lambda) \mu_{e_i \rightarrow h_j}^{t-1}(e_i). \end{aligned}$$

Здесь $\mu_{h_j \rightarrow e_i}(e_i)$ и $\mu_{e_i \rightarrow h_j}(e_i)$ – сообщения, вычисляемые на шагах 2 и 3 общего алгоритма декодирования.

Формулировка задания

1. Реализовать алгоритм построения по заданной проверочной матрице чётности H порождающей матрицы кода G для систематического кодирования;
2. Реализовать алгоритм декодирования низкоплотностного кода на основе loopy BP; при реализации шага 2 пересчёта сообщений от факторов к переменным необходимо использовать эффективные схемы, обозначенные выше; реализовать последовательное и параллельное расписание пересчёта сообщений, а также дэмпфирование сообщений; при реализации на MATLAB одной итерации схемы передачи сообщений допускается использование циклов только глубины 1; сделать иллюстративную картинку процесса декодирования (например, как на рис. 1);
3. Провести эксперименты с различными расписаниями пересчёта сообщений и коэффициентами дэмпфирования; в частности, оценить долю стабилизировавшихся beliefs в зависимости от номера итерации алгоритма декодирования (усреднённую по различным запускам); оценить время работы алгоритма декодирования в зависимости от выбранного расписания и коэффициента дэмпфирования;
4. Рассмотрим две характеристики качества кода – вероятность совершить ошибку хотя бы в одном бите при декодировании блока ($p(\hat{e} \neq e)$) и среднюю вероятность совершить ошибку при декодировании в одном бите ($\frac{1}{n} \sum_{i=1}^n p(\hat{e}_i \neq e_i)$). Требуется реализовать алгоритм оценки вероятности битовой и блоковой ошибки кода с помощью метода стат. испытаний (многократная случайная генерация вектора ошибок e ($e_i = 1$ с вероятностью q), вычисление по нему синдрома $s = He$, восстановление вектора ошибок \hat{e} с помощью алгоритма декодирования и подсчет необходимых характеристик);

5. Провести эксперименты по оцениванию битовой и блоковой ошибки низкоплотного кода для различных значений длины кодового слова n , скорости кода r , вероятности инвертирования бита при передаче по каналу связи q и среднего количества единиц в столбце проверочной матрицы j . В частности, необходимо проанализировать следующие ситуации:
 - Теорема Шеннона определяет пропускную способность канала как максимально допустимую скорость кода, при которой возможно осуществление надежной коммуникации. Требуется проверить, как меняются характеристики кода при изменении скорости r от минимального значения до пропускной способности канала (при изменении скорости r рекомендуется фиксировать длину кода n).
 - Теорема Шеннона предполагает, что качество кода растет при увеличении длины кодового слова n . Требуется проверить это предположение.
 - Одно из следствий теоремы Шеннона утверждает, что хорошими кодами являются коды со случайной проверочной матрицей H . В частности, здесь предполагается, что качество кода должно расти при увеличении среднего количества единиц в столбце проверочной матрицы j . Требуется проверить это утверждение для низкоплотных кодов (путём рассмотрения нескольких значений j , начиная от 3).
6. Провести эксперименты по сравнению низкоплотного кода с кодами БЧХ¹; в частности, рассмотреть ситуацию длинных и коротких кодов;
7. Составить отчёт в формате PDF с описанием всех проведённых исследований.

Рекомендации по выполнению задания

1. Разреженную проверочную матрицу кода заданных размеров можно строить с помощью случайной генерации (с соблюдением условия полноранговости). Однако, здесь рекомендуется воспользоваться [готовой реализацией](#), которая генерирует проверочную матрицу заданного размера с заданным количеством единиц в каждом столбце. При этом реализованный алгоритм старается сократить количество циклов длины 4 в генерируемой матрице, т.к. наличие коротких циклов в графе, как правило, усложняет работу алгоритма коду ВР.
2. При тестировании алгоритма декодирования рекомендуется пробовать запускать итерационный процесс из различных начальных приближений для сообщений (не только из значений унарных потенциалов). При корректной реализации метода должна наблюдаться сходимость к одним и тем же финальным сообщениям независимо от начальных приближений.

Оформление задания

Выполненное задание следует отправить письмом по адресу bayesml@gmail.com с заголовком письма

[ГМ15] Задание 1 <ФИО>.

Убедительная просьба присылать выполненное задание **только один раз** с окончательным вариантом. Также убедительная просьба строго придерживаться заданных ниже прототипов реализуемых функций.

¹Этот пункт задания является необязательным для слушателей не из 417-й группы

Таблица 1: Построение порождающей матрицы для систематического кодирования

$[G, \text{ind}] = \text{ldpc_gen_matrix}(H)$
<p>ВХОД</p> <p>H – проверочная матрица чётности, бинарная матрица размера $m \times n$;</p>
<p>ВЫХОД</p> <p>G – порождающая матрица кода, бинарная матрица размера $n \times k$;</p> <p>ind – номера позиций кодового слова, в которые копируются биты исходного сообщения, т.е. $G(\text{ind}, :)$ является единичной матрицей.</p>

Таблица 2: Алгоритм декодирования LDPC-кода в синдромном представлении

$[\mathbf{e}, \text{status}] = \text{ldpc_decoding}(s, H, q, \text{param_name1}, \text{param_value1}, \dots)$
<p>ВХОД</p> <p>s – наблюдаемый синдром, бинарный вектор-столбец длины m;</p> <p>H – проверочная матрица чётности, бинарная матрица размера $m \times n$;</p> <p>q – вероятность инверсии бита при передаче по каналу связи, число от 0 до 1/2;</p> <p>($\text{param_name}, \text{param_value}$) – набор необязательных параметров алгоритма, следующие имена и значения возможны:</p> <ul style="list-style-type: none"> 'schedule' – расписание пересчёта сообщений, возможные значения 'parallel' и 'sequential', по умолчанию = 'parallel'; 'damping' – коэффициент дэмпфирования при пересчёте сообщений, число от 0 до 1, по умолчанию = 1; 'max_iter' – максимальное число итераций алгоритма декодирования, число, по умолчанию = 200; 'tol_beliefs' – порог стабилизации для beliefs, число, по умолчанию = $1e - 4$; 'display' – режим отображения, true или false, если true, то отображается промежуточная информация на итерациях, например, номер итерации, текущее число ошибок декодирования, невязка для сообщений и т.д.
<p>ВЫХОД</p> <p>\mathbf{e} – восстановленный вектор ошибок, бинарный вектор-столбец длины n;</p> <p>status – результат декодирования:</p> <ul style="list-style-type: none"> 0, если найден вектор \mathbf{e}, соответствующий входному синдрому s, 1, если произошла стабилизация значений beliefs, 2, если произошёл выход по максимальному числу итераций.

Таблица 3: Оценка характеристик LDPC-кода с помощью метода Монте Карло

$[\text{err_bit}, \text{err_block}, \text{diver}] = \text{ldpc_mc}(H, q, \text{num_points})$
<p>ВХОД</p> <p>H – проверочная матрица чётности, бинарная матрица размера $m \times n$;</p> <p>q – вероятность инверсии бита при передаче по каналу связи, число от 0 до 1/2;</p> <p>num_points – общее количество стат. испытаний, число;</p>
<p>ВЫХОД</p> <p>err_bit – вероятность битовой ошибки декодирования (относительно n бит кодового слова), число от 0 до 1, вычисляется по тем ситуациям, когда алгоритм декодирования сошёлся ($\text{status} < 2$);</p> <p>err_block – вероятность блоковой ошибки декодирования, число от 0 до 1, вычисляется по тем ситуациям, когда алгоритм декодирования сошёлся ($\text{status} < 2$);</p> <p>diver – доля ситуаций расходимости алгоритма декодирования, число от 0 до 1.</p>