TUGAS AKHIR - SM 141501

KAJIAN KODE HAMMING DAN SIMULASINYA MENGGUNAKAN SAGE

TAHTA DARI TIMUR
NRP 1211 100 123

Dosen Pembimbing
Dr. Dieky Adzkiya, S.Si, M.Si
Soleha, S.Si, M.Si

DEPARTEMEN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR - SM 141501

KAJIAN KODE HAMMING DAN SIMULASINYA MENGGUNAKAN SAGE

TAHTA DARI TIMUR
NRP 1211 100 123

Dosen Pembimbing:
Dr. Dieky Adzkiya, S.Si, M.Si
Soleha, S.Si, M.Si

DEPARTEMEN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT - SM 141501

A STUDY OF HAMMING CODE AND ITS SIMULATIONS USING SAGE

TAHTA DARI TIMUR
NRP 1211 100 123

Supervisor:
Dr. Dieky Adzkiya, S.Si, M.Si
Soleha, S.Si, M.Si

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Sciences
Sepuluh Nopember Institute of Technology
Surabaya 2017

LEMBAR PENGESAHAN

KAJIAN KODE HAMMING DAN SIMULASINYA MENGGUNAKAN SAGE

A STUDY OF HAMMING CODE AND ITS SIMULATIONS USING SAGE

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana
pada Bidang Studi Analisis dan Aljabar
Program Studi S-1 Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
TAHTA DARI TIMUR
NRP. 1211 100 123

Menyetujui,

Dosen Pembimbing II,	Dosen Pembimbing I,
Soleha, S.Si, M.Si	Dr. Dieky Adzkiya, S.Si, M.Si
NIP. 19830107 200604 2 001	NIP. 19830517 200812 1 003

Mengetahui,
Ketua Departemen Matematika
FMIPA ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, 19 Juli 2017

KAJIAN KODE HAMMING DAN SIMULASINYA MENGGUNAKAN SAGE

Nama Mahasiswa	:	Tahta Dari Timur
NRP	:	1211 100 123
Departemen	:	Matematika FMIPA-ITS
Pembimbing	:	1. Dr. Dieky Adzkiya, S.Si, M.Si
		2. Soleha, S.Si, M.Si

Abstrak

Transmisi data digital melalui noisy channel dapat mengakibatkan perubahan pada data yang sedang ditransmisikan. Untuk menjamin integritas data selama transmisi, dikembangkanlah teori koding untuk proses encoding dan decoding sebelum transmisi. Data akan diencode menjadi kode, ditransmisikan melalui suatu channel, dan diberi error. Penerima akan memeriksa apakah terdapat error dan bila ada memperbaikinya. Kode yang diterima ini akan didecode kembali menjadi data seperti semula. Dalam tugas akhir ini akan dibahas suatu bentuk kode yang kode Hamming. Kode Hamming adalah kode linear yang memiliki laju transmisi yang besar dan metode algoritma decoding biner yang cepat. Proses simulasi dilakukan untuk mengetahui efisiensi metode-metode encoder/decoder yang digunakan. Selain itu dapat diketahui sifat-sifat kode seperti matriks cek, matriks generator, error yang terjadi, kecepatan (information rate), radius error-detecting dan radius error-correcting. Hasil rancangan ini kemudian akan diimplementasikan dan dijalankan sebagai simulasi proses koding data digital (teks dan gambar) menggunakan Sage.

Kata-kunci: *data digital, encoding, decoding, error-correcting, kode linear, kode Hamming, simulasi, SageMath*

A STUDY OF HAMMING CODE AND ITS SIMULATIONS USING SAGE

Name	:	Tahta Dari Timur
NRP	:	1211 100 123
Department	:	Mathematics FMIPA-ITS
Supervisors	:	1. Dr. Dieky Adzkiya, S.Si, M.Si
		2. Soleha, S.Si, M.Si

Abstract

Digital data transmission through a noisy channel could alter the data being transmitted. To ensure data integrity, coding theory developed to encode information prior to transmission. Data will be encoded to a codeword, transmitted through a noisy channel, and received errors. Receiver must check whether there are errors in codewords received and correct them. After being corrected, this codeword will be decoded back to the original message. In this final project, we will discuss a type of code called the Hamming code. Hamming code is a linear code which has a property of large transmission rate and fast binary decoding method. Furthermore, a system of simulation will be designed to compare their encode/decode methods. Moreover we could know their properties e.g. parity check and generator matrix, errors, information rate, error-detecting and error-correcting radiuses. This system of simulation will be implemented and run on Sage using text and image as data sample.

Key-words: *digital data, encode, decode, error-correcting, linear code, Hamming code, simulation, SageMath*

KATA PENGANTAR

Puji dan syukur ke hadirat Allah SWT yang telah memberikan rahmat kepada penulis sehingga dapat menyelesaikan penulisan buku laporan Tugas Akhir ini dengan judul **"Kajian Kode Hamming dan Simulasinya Menggunakan Sage"**. Buku laporan Tugas Akhir ini ditulis sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi S-1 Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam di Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa tanpa dukungan serta bimbingan, penulisan buku laporan Tugas Akhir ini tidak akan berjalan dengan lancar. Oleh karena itu, izinkan penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Imam Mukhlash, S.Si, MT selaku Ketua Departemen Matematika FMIPA ITS,

2. Bapak Dr. Dieky Adzkiya, S.Si, M.Si dan Ibu Soleha S.Si, M.Si selaku dosen pembimbing tugas akhir,

3. Ibu Dian Winda S., S.Si, M.Si sebagai dosen wali penulis selama masa studi di Departemen Matematika FMIPA ITS,

4. Ibu Dr. Dwi Ratna S., S.Si, MT, Bapak Drs. Sadjidon, M.Si dan Ibu Dian Winda S., S.Si, M.Si sebagai dosen penguji tugas akhir,

5. Bapak Dr. Didik Khusnul Arif, S.Si, M.Si selaku Kaprodi S-1 Departemen Matematika FMIPA ITS,

6. Bapak Drs. Iis Herisman, M.Si selaku Sekprodi S-1 Departemen Matematika FMIPA ITS,

7. Ayahanda, Ibunda dan seluruh keluarga tercinta,

8. Seluruh pihak yang secara langsung atau tidak telah memberikan dukungan kepada penulis.

Penulis berharap buku laporan ini dapat mencapai tujuannya dan membawa manfaat baik untuk penulis sendiri ataupun orang lain.

Surabaya, 18 Juli 2017

Tahta Dari Timur

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Komunikasi digital adalah metode komunikasi yang banyak digunakan dan berkembang pesat dalam beberapa dekade terakhir. Pengembangan perangkat-perangkat komunikasi elektronik, teknologi nirkabel dan internet memungkinkan manusia berkomunikasi dalam jarak jauh dan dalam waktu singkat. Data-data digital ini dikirimkan melalui suatu saluran (*channel*) yang mungkin saja *noisy* (dapat merusak pesan). Akibatnya, kesalahan atau error mungkin saja terjadi selama proses transmisi data melalui saluran ini.

Sebagai contoh, misalkan seorang A hendak mengirimkan pesan **butuh** kepada seorang B menggunakan saluran *noisy*. Setelah itu, B menerima pesan **bunuh**, yang sudah mengalami perubahan makna dari pesan aslinya. Atau mungkin saja B menerima suatu pesan yang sama sekali tidak bermakna seperti **hydpq**. Misalkan saluran yang digunakan memiliki peluang mengirimkan pesan secara benar $p = 0.99$, dan peluang mengirim pesan secara salah $q = 1 - p = 0.01$. Jika seorang pengirim A mengirim pesan sepanjang 1000 karakter, maka penerima B masih dapat menerima pesan dengan 10 kesalahan, yang dapat berakibat fatal. Dalam sebuah komputer terjadi banyak sekali proses transfer data, dan tidak jarang prosesor melakukan kesalahan komputasi. Bahkan dengan $p = 0.99$ seperti contoh sebelumnya, komputer masih dapat melakukan kesalahan atau berhenti beroperasi sebanyak 10 kali setiap 1000 operasi (Lidl dan Pilz, 1998).

Karena itu, perlu dikembangkan suatu metode untuk

mendeteksi suatu kesalahan (*error*) yang terjadi dalam suatu transmisi data dan jika mungkin memperbaikinya. Latar belakang inilah yang mendasari pengembangan teori koding (*coding theory*). Pada tahun 1950 hingga 1990an secara terpisah, Claude E. Shannon dan Richard W. Hamming mulai mempelajari dan mengembangkan metode yang dimaksud (Neubauer, Freudenberger dan Kühn, 2007). Dengan kata lain, teori koding menjamin keutuhan/integritas transmisi data digital, dan memperbaiki kesalahan-kesalahan yang mungkin terjadi.

Sebagai contoh, misal suatu pesan direpresentasikan dalam biner sebagai 001. Agar error dapat dideteksi, pesan diulang sebanyak dua kali menjadi 001001. Pesan ini selanjutnya ditransmisikan dan diberi error sehingga berubah menjadi 101101. Penerima akan mengalami kesulitan dalam mengkoreksi error ini karena tidak ada cara pasti untuk mengetahui kode aslinya. Kesulitan koreksi ini tetap berlaku pada pengulangan sebanyak berapa kali pun karena sifat error yang probabilistik. Jika tidak, dibutuhkan pengulangan pesan yang sangat banyak sehingga berakibat panjang pesan bertambah berkali-kali lipat (Lidl dan Pilz, 1998). Artinya, diperlukan suatu metode encoding tertentu yang memberikan *check bit* yang cukup pada kode untuk pendeteksi error, tetapi tidak terlalu panjang sehingga beban transmisi menjadi berlebih. Di sisi lain, juga diperlukan metode decoding tertentu untuk dapat mengkoreksi error yang terjadi pada kode dan untuk mendecode kode menjadi pesan semula. Misal suatu pesan $\mathbf{a} \in \mathbb{F}_q^m$ dan kode $\mathbf{b} \in \mathbb{F}_q^n$. Pesan $\mathbf{a}$ akan diencode menjadi kode $\mathbf{b}$, ditransmisikan dan diberi error $\mathbf{e} \in \mathbb{F}_q^n$ sehingga menjadi kode $\mathbf{c} \in \mathbb{F}_q^n$. Kode $\mathbf{c}$ selanjutnya akan dikoreksi menjadi $\mathbf{b}$, dan akhirnya didecode menjadi pesan $\mathbf{a}$ kembali. Secara garis besar, keseluruhan proses pengiriman pesan $\mathbf{a}$ adalah:

$$\mathbf{a} \xrightarrow{\text{encode}} \mathbf{b} \xrightarrow{\text{noise}} \mathbf{c} \xrightarrow{\text{koreksi}} \mathbf{b} \xrightarrow{\text{decode}} \mathbf{a}$$

Dapat dikatakan bahwa salah satu tantangan dalam teori koding adalah bagaimana mengembangkan metode encoding/decoding sehingga kode tidak terlalu panjang, namun cukup untuk dapat dilakukan koreksi dan didecode secara benar menjadi pesan asli. Dengan kata lain, selain kebenaran metode encoding/decoding yang digunakan, panjang kode yang dihasilkan yaitu n, harus memiliki panjang yang cukup sehingga laju informasi m/n cukup besar.

Salah satu contoh penggunaan teori koding adalah pada sistem ISBN (*International Standard Book Number*). Pada ISBN-10, digit kesepuluh adalah digit cek yang diperoleh dengan menghitung $d_{10} = 11 - [10d_1 + 9d_2 + \cdots + 2d_9 \,(\mathrm{mod} 11)]$ sehingga kesalahan penulisan/pembacaan kode ISBN pada digit pertama hingga kesembilan dapat dideteksi. Kode ISBN merupakan salah satu contoh kode *error-detecting*, yang artinya kode ISBN dapat mendeteksi apakah terdapat kesalahan pada digit-digit di kodenya, tetapi tidak dapat memperbaiki kesalahan yang terjadi (Lidl dan Pilz, 1998). Pada tahun 1960an NASA meluncurkan pesawat nirawak Mariner 9 ke Mars untuk mengirimkan foto-foto Mars. Foto-foto hitam putih ini dikirimkan dari Mars ke Bumi melalui ruang angkasa menggunakan kode Reed-Muller dengan panjang $n = 32$, dengan bit informasi sepanjang $k = 6$ bit dan bit pemeriksaan (*check bit*) sepanjang $n - k = 26$ bit (Joyner dan Kim, 2011). Beberapa contoh aplikasi kode Hamming lainnya adalah pada steganografi video aman menggunakan kode Hamming (7,4) (Mstafa dan Elleithy, 2010), kompresi data lossless (Al-Bahadili, 2007), dan otentikasi watermark citra medis (Abadi, 2010).

Berdasarkan latar belakang di atas serta kajian atas

beberapa sumber, akan dikaji teori dasar dari kode Hamming beserta algoritma-algoritma encoding dan decodingnya. Kode Hamming adalah kode linear yang memiliki laju transmisi besar dan metode decoding biner yang cepat. Laju transmisi adalah perbandingan antara panjang pesan k dan panjang *codeword* n. Dengan kata lain, kode Hamming mentransmisikan lebih banyak informasi k dalam setiap *codeword* n (Lidl dan Pilz, 1998). Algoritma encoding yang akan dikaji yaitu encoder *Matriks Generator* dan *Parity Check*. Algoritma decoding yang akan dikaji adalah decoder *Nearest Neighbor Decoding* dan *Syndrome*. Langkah-langkah sistematis akan dibuat untuk melakukan simulasi encoder/decoder tersebut, diantaranya error pada saluran, laju transmisi informasi (information rate), radius error-detecting dan radius error-correcting.

Langkah-langkah tersebut akan diimplementasikan menggunakan perangkat lunak SageMath (Developers, 2017) di mana akan dilakukan simulasi pengiriman data digital (teks dan gambar). Proses pengerjaan simulasi ini akan menggunakan modul-modul Interact(Stein dan Grout, 2017), Coding Theory (Joyner, Stein, Alexander dan Johnson, 2017), OpenCV Python (Team, 2017*a*), dan beberapa modul lainnya oleh para pengembang dan kontributor perangkat lunak bebas SageMath yang terlibat baik dalam versi-versi terdahulu maupun versi yang akan digunakan dalam penulisan ini.

1.2 Rumusan Masalah

Permasalahan yang akan diselesaikan dalam tugas akhir ini adalah:

1. Bagaimana perancangan proses simulasi dari metode-metode encoder/decoder kode Hamming berdasarkan kajian teori,

2. Bagaimana implementasi simulasi dari metode-metode encoding/decoding kode Hamming menggunakan Sage,

3. Bagaimana perbandingan antara algoritma encoder dan decoder kode Hamming dalam hal waktu komputasi berdasarkan hasil simulasi.

1.3 Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah:

1. Kode yang akan dibahas adalah kode Hamming,

2. Encoder yang akan dibahas/digunakan adalah Matriks Generator dan *Parity Check*,

3. Decoder yang akan dibahas/digunakan adalah Nearest Neighbor dan Syndrome,

4. Lapangan berhingga yang akan digunakan adalah $\mathbb{F}_2$ (biner),

5. Bahasa pemrograman yang digunakan adalah bahasa Python/Sage.

1.4 Tujuan

Tujuan dari penulisan tugas akhir ini adalah:

1. Merancang proses simulasi metode-metode encoder/decoder kode Hamming berdasarkan kajian teori,

2. Melakukan simulasi kode Hamming dalam mengirimkan dua tipe data yaitu teks dan gambar.

1.5 Manfaat

Adapun manfaat dari tugas akhir ini adalah:

1. Memberikan pengetahuan dasar tentang teori koding dan aplikasinya,

2. Bagi pembaca secara umum, tugas akhir ini diharapkan dapat membantu dalam memperkenalkan dan memudahkan pemahaman dasar-dasar teori koding dan menyediakan program simulasi sebagai alat pendukung.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas beberapa teori dan konsep matematika yang mendasari teori koding.

2.1 Koset Dari Grup

Pada subbab ini akan dibahas grup, subgrup, orde dari grup dan elemen grup, koset, beserta beberapa contohnya.

Sebuah himpunan takkosong G bersama dengan operasi biner $* : G \times G \to G$ disebut **grup** jika kondisi-kondisi berikut terpenuhi:

1. Operasi $*$ asosiatif di G. Dengan kata lain selalu berlaku $a * (b * c) = (a * b) * c$ untuk semua $a, b, c \in G$,

2. Terdapat elemen identitas $e \in G$ sehingga $a*e = e*a = a$ untuk semua $a \in G$, dan

3. Untuk semua $a \in G$, terdapat elemen invers $a^{-1} \in G$ sehingga $a * a^{-1} = a^{-1} * a = e$.

Jika G memenuhi kondisi-kondisi di atas, maka G adalah grup terhadap operasi $*$ dan ditulis $(G, *)$. Jika untuk sebarang $a, b \in G$ berlaku kondisi $a * b = b * a$ (komutatif), maka $(G, *)$ disebut **grup abel** (Pinter, 1982; Rotman, 2002).

Misal diberikan suatu himpunan $H \subseteq G$ dengan $(G, *)$ grup dan $a, b \in H$. Jika $a, b \in H$ berakibat $a * b \in H$ dan jika $a \in H$ berakibat $a^{-1} \in H$, maka H disebut **subgrup** dari G (Pinter, 1982; Rotman, 2002). Himpunan $\{e\}$ dan G adalah subgrup dari grup G. Subgrup-subgrup ini disebut *subgrup*

trivial. Subgrup selain subgrup tersebut disebut **subgrup sejati** (*proper subgroup*).

Orde dari suatu grup G adalah banyaknya elemen di G, ditulis $|G|$. Orde dari suatu elemen $a \in G$ atau ditulis ditulis $|a|$ adalah bilangan bulat positif terkecil n sedemikian hingga:

$$a^n = \underbrace{a * a * \ldots * a}_{\text{n kali}} = e$$

Grup dengan orde berhingga disebut grup berhingga.

Misal diberikan grup $(G, *)$, subgrup dari $(G, *)$ yaitu $(H, *)$ dan $a \in G$. **Koset** dari G adalah himpunan bagian aH dan Ha (Pinter, 1982; Rotman, 2002), dengan

$$aH = \{a * h : h \in H\}$$

$$Ha = \{h * a : h \in H\}$$

Koset aH disebut **koset kiri**, sedangkan Ha disebut **koset kanan** dari G.

Contoh 2.1.1. Himpunan $\mathbb{Z}_6$ adalah grup atas operasi penjumlahan modulo 6. Himpunan $H = \{0, 2, 4\}$ adalah subgrup dari $\mathbb{Z}_6$. Orde grup $|\mathbb{Z}_6| = 6$ dan orde dari setiap elemennya adalah:

$$|0| = 0$$

$$|1| = 6$$

$$|2| = 3$$

$$|3| = 2$$

$$|4| = 3$$

$$|5| = 6$$

Koset-koset kiri dari H untuk $a \in \mathbb{Z}_6$ adalah:

$$0 + H = \{(0 + h) \bmod 6 : h \in H\} = \{0, 2, 4\}$$
$$1 + H = \{(1 + h) \bmod 6 : h \in H\} = \{1, 3, 5\}$$
$$2 + H = \{(2 + h) \bmod 6 : h \in H\} = \{2, 4, 0\}$$
$$3 + H = \{(3 + h) \bmod 6 : h \in H\} = \{3, 5, 1\}$$
$$4 + H = \{(4 + h) \bmod 6 : h \in H\} = \{4, 0, 2\}$$
$$5 + H = \{(5 + h) \bmod 6 : h \in H\} = \{5, 1, 3\}$$

Koset kanan dapat diperoleh dengan cara yang sama. ■

2.2 Lapangan Berhingga

Pada subbab ini akan dibahas definisi ring, subring, pembagi nol dan lapangan, serta beberapa contohnya.

Sebuah himpunan takkosong R bersama dengan dua operasi biner $+ : R \times R \to R$ (penjumlahan) dan $\cdot : R \times R \to R$ (perkalian) disebut **ring** jika kondisi-kondisi berikut terpenuhi:

1. $(R, +)$ adalah grup abel,
2. Operasi perkalian asosiatif di R. Dengan kata lain selalu berlaku $r \cdot (s \cdot t) = (r \cdot s) \cdot t$ untuk semua $r, s, t \in R$,
3. Operasi perkalian distributif atas penjumlahan. Artinya selalu berlaku $r \cdot (s+t) = r \cdot s + r \cdot t$ dan $(r+s) \cdot t = r \cdot t + r \cdot s$ untuk semua $r, s, t \in R$.

Jika untuk semua $r, s \in R$ berlaku $r \cdot s = s \cdot r$ maka R disebut **ring komutatif**. Jika terdapat elemen $1 \in R$ sehingga berlaku $1 \cdot r = r \cdot 1 = r$, maka R disebut **ring satuan** (Pinter, 1982; Rotman, 2002). Sebagai catatan penulisan, operasi perkalian dua elemen cukup ditulis sebagai rs untuk tujuan penyederhanaan. Simbol 0 dan $-r$ berturut-turut menyatakan elemen netral R terhadap penjumlahan dan

elemen invers penjumlahan, sedangkan elemen invers terhadap perkalian ditulis r^{-1}.

Suatu himpunan bagian $S \neq \emptyset$ dari ring R disebut **subring** dari R jika S sendiri adalah ring atas operasi-operasi pada R. Misal diberikan suatu ring R dan $r, s \in R$. Jika $r, s \neq 0$ dan berlaku $rs = 0$, maka r dan s disebut **pembagi nol** (Pinter, 1982; Rotman, 2002).

Contoh 2.2.1. Himpunan bilangan bulat $\mathbb{Z}$ adalah ring terhadap operasi penjumlahan dan perkalian biasa. $\mathbb{Z}$ juga merupakan ring komutatif dan ring satuan. Himpunan bagian $2\mathbb{Z}$ adalah subring dari $\mathbb{Z}$. ■

Contoh 2.2.2. Himpunan $\mathbb{Z}_4$ adalah ring terhadap operasi penjumlahan dan perkalian modulo 4. Berikut tabel operasinya.

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$*$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

$\mathbb{Z}_4$ adalah ring komutatif dan ring satuan. Himpunan $\{0, 2\}$ adalah subring dari $\mathbb{Z}_4$. ■

Suatu ring F disebut **lapangan** jika (Pinter, 1982; Rotman, 2002):

1. F adalah ring komutatif,

2. F memuat elemen satuan $1 \in F$,

3. Setiap elemen taknol di F memiliki invers terhadap operasi perkalian.

Lapangan berhingga $\mathbb{F}$ adalah suatu lapangan yang banyak elemennya berhingga. Misal $\mathbb{F}$ memiliki elemen satuan 1. Bilangan bulat positif terkecil p yang memenuhi $q \cdot 1 = 0$ disebut **karakteristik** dari $\mathbb{F}$, ditulis kar($\mathbb{F}$). Selanjutnya, lapangan berhingga $\mathbb{F}$ dengan kar($\mathbb{F}$)$= q$ akan ditulis $\mathbb{F}_q$.

Contoh 2.2.3. Himpunan $\mathbb{R}$ dan $\mathbb{C}$ adalah contoh lapangan, sedangkan $\mathbb{Z}$ tidak, sebab elemen-elemen $\mathbb{Z}$ tidak selalu memiliki invers terhadap perkalian. $\mathbb{R}$ dan $\mathbb{C}$ adalah dua contoh lapangan tak berhingga. ■

Contoh 2.2.4. Ring $\mathbb{Z}_2$ dan $\mathbb{Z}_3$ adalah lapangan berhingga dengan karakteristik 2 dan 3. Selanjutnya dua himpunan ini ditulis $\mathbb{F}_2$ dan $\mathbb{F}_3$. Sage melambangkan lapangan berhingga karakteristik q dengan GF(q)(Team, 2017*b*). ■

2.3 Ruang Vektor

Pada subbbab ini akan dibahas ruang vektor dan beberapa sifat matriks. Suatu matriks A berukuran $m \times n$ ditulis sebagai:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Vektor adalah matriks berukuran $m \times 1$, ditulis:

$$\mathbf{x} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

atau

$$\mathbf{x} = (b_1, b_2, \cdots, b_m)$$

Jika $b_i \in V$ dengan V sebarang himpunan tak kosong, maka $\mathbf{x} \in V^m$. Kolom-kolom dari A dapat dipandang sebagai vektor $\mathbf{a}_i$, sehingga A dapat ditulis sebagai $\begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{pmatrix}$

Definisi 2.3.1 (Ruang Vektor)**.** (Lay, Lay dan McDonald, 2016) Himpunan tak kosong $\mathcal{V}$ bersama operasi penjumlahan dan perkalian skalar disebut **ruang vektor** atas lapangan K, jika untuk *vektor* $\mathbf{u}, \mathbf{v} \in \mathcal{V}$ dan *skalar* $c, d \in K$ berlaku:

1. $(\mathcal{V}, +)$ grup abel,
2. Perkalian skalar $c\mathbf{u} \in \mathcal{V}$,
3. $c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v}$,
4. $(c + d)\mathbf{u} = c\mathbf{u} + d\mathbf{u}$,
5. $c(d\mathbf{u}) = (cd)\mathbf{u}$,
6. $1\mathbf{u} = \mathbf{u}$, dengan $1 \in K$.

Misal diberikan $\mathcal{V}$ ruang vektor atas lapangan K dan $\mathcal{H}$ himpunan bagian dari $\mathcal{V}$. $\mathcal{H}$ disebut **subruang vektor** (Lay dkk., 2016) dari $\mathcal{V}$ atas lapangan K jika:

1. Untuk setiap $\mathbf{u}, \mathbf{v} \in \mathcal{H}$ berlaku $\mathbf{u} - \mathbf{v} \in \mathcal{H}$,
2. $\mathcal{H}$ tertutup atas perkalian dengan skalar, yaitu $c\mathbf{u} \in \mathcal{H}$ untuk semua $\mathbf{u} \in \mathcal{H}$ dan $c \in K$.

Diberikan himpunan vektor $U = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$. Himpunan $\langle U \rangle$ adalah himpunan semua vektor $\mathbf{v}$ yang merupakan kombinasi linear vektor-vektor di U, yaitu:

$$\langle U \rangle = \{\mathbf{v} : \mathbf{v} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \ldots + c_n\mathbf{u}_n\}$$

dengan c_i adalah elemen lapangan K. Dalam hal ini dikatakan bahwa himpunan $\langle U \rangle$ dibentangkan/dibangkitkan oleh U. Himpunan vektor-vektor $U = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ dikatakan **bebas linear** jika persamaan:

$$c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \ldots + c_n\mathbf{u}_n = \mathbf{0}$$

hanya memiliki solusi trivial, yaitu $c_1 = c_2 = \ldots = c_n = 0$ (Lay dkk., 2016). Himpunan $\mathcal{B} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ disebut **basis** untuk ruang vektor $\mathcal{V}$ jika (Lay dkk., 2016):

1. Setiap elemen $\mathcal{B}$ bebas linear,
2. $\mathcal{B}$ membentang $\mathcal{V}$.

Suatu ruang vektor $\mathcal{V}$ atas lapangan K dikatakan memiliki dimensi n, ditulis $\dim(\mathcal{V}) = n$, jika $\mathcal{V}$ dibentangkan oleh suatu basis $\mathcal{B}$ dengan banyak elemen berhingga n (Lay dkk., 2016). Jika elemen $\mathcal{B}$ takhingga, maka dikatakan $\mathcal{V}$ berdimensi takhingga. Himpunan $\{\mathbf{0}\}$ memiliki dimensi $\dim(\{\mathbf{0}\}) = 0$ (Lay dkk., 2016).

Contoh 2.3.1. $\mathbb{R}^3$ adalah contoh ruang vektor atas lapangan $\mathbb{R}$. Himpunan

$$V = \left\{ \begin{pmatrix} s \\ t \\ 0 \end{pmatrix} : s, t \in \mathbb{R} \right\}$$

adalah subruang dari $\mathbb{R}^3$. Tetapi, $\mathbb{R}^2$ *bukan* merupakan subruang dari $\mathbb{R}^3$ karena $\mathbb{R}^2$ bukanlah himpunan bagian dari $\mathbb{R}^3$. ■

Contoh 2.3.2. Himpunan

$$V = \left\{ \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} : a, b, c, d \in \mathbb{F}_3 \right\}$$

adalah contoh ruang vektor atas lapangan $\mathbb{F}_3$. V memiliki basis $\mathcal{B} = [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)]$ dan dimensi 4. Contoh subruang dari V adalah ruang vektor berdimensi 3 dengan basis $\mathcal{B} = [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0)]$. ■

Pembahasan selanjutnya adalah mengenai beberapa sifat matriks.

Ruang nol (kernel) dari suatu matriks A berukuran $m \times n$ atas lapangan K, ditulis nol(A), adalah himpunan semua solusi dari persamaan $A\mathbf{x}^T = \mathbf{0}$ (Lay dkk., 2016) atau

$$\text{nol}(A) = \{\mathbf{x} : \mathbf{x} \in K^n, A\mathbf{x} = \mathbf{0}\}$$

Dengan kata lain, ruang nol adalah semua vektor $\mathbf{x} \in K^n$ yang dipetakan oleh matriks A ke $\mathbf{0} \in K^m$. Ruang kolom dari suatu matriks A berukuran $m \times n$ atas lapangan K, ditulis kol(A), adalah himpunan semua kombinasi linear yang dibentangkan oleh kolom A (Lay dkk., 2016). Jika $A = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{pmatrix}$, maka

$$\text{kol}(A) = \langle \mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n \rangle$$

Rank dari suatu matriks A adalah dimensi ruang kolom A (Lay dkk., 2016), yaitu:

$$\text{rank}(A) = \dim(\text{kol}(A))$$

Contoh 2.3.3. Matriks M atas $\mathbb{F}_2$

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

memiliki nol(M)=$\langle (1, 1, 1, 0, 1, 1), (0, 0, 0, 1, 1, 0) \rangle$. M dapat diubah menjadi bentuk eselon, yaitu

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

sehingga kol(R)=$\langle (1,\,0,\,0,\,0)\,, (0,\,1,\,0,\,0)\,, (0,\,0,\,1,\,0)\,, (0,\,0,\,0,\,1)\,\rangle$. Dari sini diperoleh rank(M)=dim(kol(R))=4. ■

2.4 Ruang Metrik

Pada subbab ini akan didefinisikan ruang metrik dan ruang bernorma.

Definisi 2.4.1 (Ruang Metrik)**.** (Kreyszig, 1978) Ruang metrik adalah pasangan (X, d) dengan X himpunan tak kosong dan d metrik (atau dapat disebut fungsi jarak) pada X. Fungsi d adalah fungsi yang didefinisikan di $X \times X \to \mathbb{R}$ sehingga jika $x, y, z \in X$, berlaku:

1. d bernilai real, berhingga dan taknegatif,
2. $d(x, y) = 0$ jika dan hanya jika $x = y$,
3. $d(x, y) = d(y, x)$,
4. $d(x, y) \leq d(x, z) + d(z, y)$

Contoh 2.4.1. Himpunan bilangan real $\mathbb{R}$ bersama dengan $d(x, y) = |x - y|$ adalah ruang metrik. ■

Contoh 2.4.2. (Kreyszig, 1978) Sebarang himpunan X dan metrik diskrit:

$$d(x, y) = \begin{cases} 0, & \text{jika } x = y, \\ 1, & \text{jika } x \neq y. \end{cases}$$

Ruang metrik ini disebut ruang metrik diskrit. ■

Subruang metrik bisa diperoleh jika terdapat $Y \subset X$ dan $\hat{d}$ terdefinisi pada Y(Kreyszig, 1978). Terdapat himpunan bagian dari ruang metrik X yang didefinisikan sebagai berikut.

Definisi 2.4.2 (Bola)**.** (Kreyszig, 1978) Diberikan titik $x_0 \in X$ dan bilangan real $r > 0$. Didefinisikan tiga macam himpunan bagian:

1. $B(x_0, r) = \{x \in X : d(x, x_0) < r\}$,
2. $\hat{B}(x_0, r) = \{x \in X : d(x, x_0) \leq r\}$,
3. $S(x_0, r) = \{x \in X : d(x, x_0) = r\}$,

Pada ketiga kasus di atas, r disebut jari-jari.

Definisi 2.4.3 (Ruang Bernorma)**.** (Kreyszig, 1978) Ruang bernorma adalah ruang vektor X bersama dengan norma yang terdefinisi di dalamnya. Di sini, norma (yang dinotasikan $\|x\|$) pada ruang vektor X adalah fungsi bernilai real dengan sifat-sifat:

1. $\|x\| \geq 0$,
2. $\|x\| = 0$ jika dan hanya jika $x = 0$,
3. $\|\alpha x\| = |\alpha|\|x\|$,
4. $\|x + y\| \leq \|x\| + \|y\|$

di mana $x, y \in X$ dan α adalah sebarang skalar.

Contoh 2.4.3. Ruang vektor $\mathbb{F}_2^3$ dan metrik $d(\mathbf{x}, \mathbf{y})$ yang menyatakan banyaknya elemen vektor $\mathbf{x}$ dan $\mathbf{y}$ yang berbeda adalah contoh ruang bernorma, dengan normanya adalah $\|\mathbf{x}\| = d(\mathbf{x}, \mathbf{0})$. ■

Konsep ruang metrik dan ruang bernorma ini digunakan untuk mendefinisikan Jarak Hamming dan Bobot Hamming pada pembahasan selanjutnya.

BAB III
METODE PENELITIAN

3.1 Tahapan Penelitian

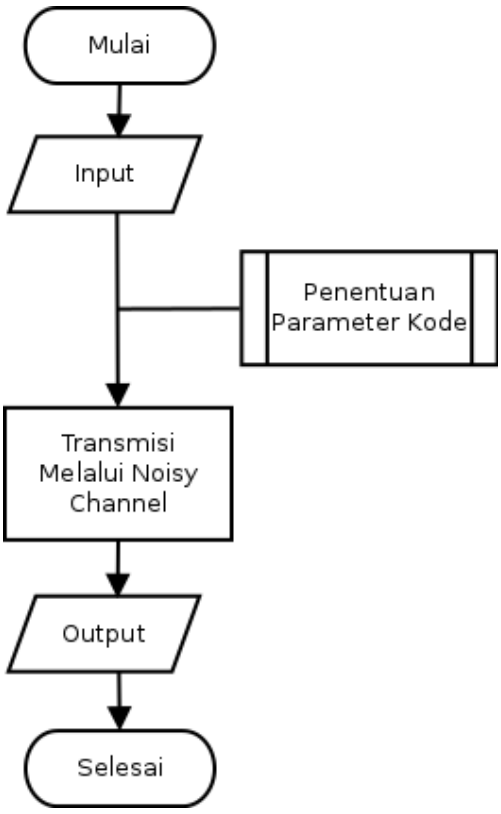
Dalam pengerjaan tugas akhir ini terdapat beberapa tahapan, yaitu:

1. Studi Literatur
   Pada tahap ini akan dipelajari dasar-dasar teori yang digunakan dalam pengkajian kode linear dan kode Hamming, yaitu grup, lapangan berhingga, ruang vektor, teorema-teorema batas (*Hamming Bound dan Plotkin Bound*), pendefinisian kode linear dan kode Hamming, dan metode-metode encoding/decoding (Matriks Generator, *Parity Check*, *Nearest Neighbor Decoding*, Syndrome),

2. Perancangan Simulasi
   Menggunakan hasil kajian teori pada langkah sebelumnya, akan dirancang suatu sistem simulasi kode linear dan kode Hamming dalam mentransmisikan data digital (teks dan gambar),

3. Implementasi Program Simulasi
   Pada tahap ini diimplementasikan program simulasi kode linear dan kode Hamming dengan Sage. Program dibuat dengan fitur *interact* dalam Sage sebagai antarmuka grafis (GUI). Pengguna dapat memberikan input berupa data digital (teks/gambar) untuk diproses, menentukan error, memilih jenis kode, dan encoder/decoder yang akan digunakan. Saat simulasi

dimulai, program menentukan parameter-parameter kode (matriks cek, matriks generator, error yang terjadi, kecepatan (information rate), radius error-detecting dan radius error- correcting) berdasarkan input yang diberikan. Output yang dihasilkan adalah beberapa parameter kode yang sedang digunakan, data sebelum dikirim, data yang mengandung error (sebelum decoding), data setelah diterima (setelah decoding) dan pernyataan True/False hasil dari pemeriksaan apakah data sebelum dikirim sama dengan data setelah diterima. Diagram proses dari program yang dimaksud ditunjukkan pada gambar berikut:



Gambar 3.1: Diagram alur program

Berikut ini adalah penjelasan dari setiap langkah pada diagram di atas.

- Setelah program dimulai, pengguna memasukkan input yaitu data yang akan ditransmisikan, error pada channel, memilih kode dan metode encoder/decodernya,
- Program secara otomatis menentukan parameter-parameter kode yang dipilih pengguna pada langkah sebelumnya,
- Data yang telah diencode ditransmisikan melalui channel dengan peluang error yang telah ditentukan pengguna pada input. Proses ini terdiri dari proses konversi data digital menjadi bentuk biner, bentuk biner diencode menjadi kode, channel memberi error secara acak pada kode, kode yang mengandung error diterima dan didecode menjadi bentuk biner, bentuk biner dikonversi kembali menjadi bentuk data awal.
- Output-output terdiri dari data awal sebelum dikirim, data yang mengandung error, data yang telah diterima, dan pernyataan True/False tentang kesamaan data sebelum dikirim dan data setelah diterima.

4. Uji Coba, Simulasi, dan Penarikan Kesimpulan
   Pada tahap ini dijalankan simulasi atas beberapa permasalahan dan contoh kasus, dalam hal ini simulasi transmisi data digital tipe teks dan gambar melalui channel dengan peluang error variatif. Terakhir diambil kesimpulan mengenai efisiensi metode-metode encoding/decoding pada kode linear dan kode Hamming.

3.2 Diagram Alur

Gambar 3.2: Diagram alur metode penelitian

3.3 Peralatan

Berikut adalah peralatan berupa perangkat keras/lunak komputer yang akan digunakan dalam penyusunan tugas akhir ini.

1. Pembuatan program simulasi menggunakan Sage versi

7.5.1 untuk Debian 8.7.1 x86-64[1],

2. Program Sage 7.5.1 dijalankan di Debian 8.7.1 64 bit atas VMWare Fusion pada komputer MacBook Pro Retina 13-inch, Intel Core i5 2.7 GHz, memori 8 GB dengan sistem operasi macOS Sierra 10.12.3 dengan kernel Darwin versi 16.3.0[2]. Perbedaan spesifikasi sistem, platform komputer, dan versi program mungkin dapat menghasilkan keluaran/hasil yang berbeda.

[1]The Debian GNU/Linux Project oleh Debian GNU/Linux Developers. http://www.debian.org

[2]VMWare, VMWare Fusion, MacBook Pro, Darwin, Intel Core, termasuk semua karya, produk, dan merk lainnya yang disebutkan/digunakan pada tulisan ini berhak cipta dan hak cipta dipegang oleh pemiliknya yang sah

BAB IV
PEMBAHASAN

4.1 Kode Linear

Pada subbab ini akan dibahas kode Hamming yang akan dimulai dengan pembahasan kode linear.

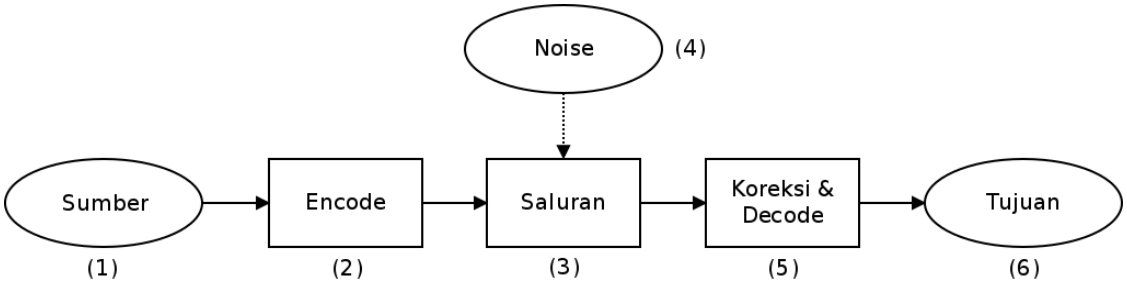
4.1.1 Kode error-detecting dan error-correcting

Transmisi data digital melalui saluran *noisy* dapat mengakibatkan error pada data/pesan. Salah satu cara untuk mendeteksi error yang terjadi adalah dengan menambahkan digit cek. Sebagai contoh, misal diberikan pesan biner yaitu $x_1x_2 \ldots x_k$. Digit cek x_{k+1} dapat ditambahkan pada pesan ini dengan $x_{k+1} = x_1 + x_2 + \ldots + x_k$. Dengan kata lain:

$$x_{k+1} = \begin{cases} 1 & \text{jika banyaknya 1 ganjil,} \\ 0 & \text{jika banyaknya 1 genap.} \end{cases}$$

Misal diberikan pesan 011. Pesan ini dapat dikodekan menjadi 0110. Artinya jika terjadi perubahan 1110 pada kode, dapat diketahui bahwa telah terjadi error pada kode ini sebab $1 + 1 + 1 \neq 0$. Namun penambahan satu digit cek ini tidak dapat mendeteksi jika terjadi dua error. Misal kode berubah menjadi 1010. Kode ini jelas tidak sama dengan kode asli, tetapi digit cek tidak dapat mendeteksinya sebab $1 + 1 = 0$. Bahkan cara ini juga tidak berlaku untuk semua jumlah error yang genap. Artinya, perlu ditambahkan lagi digit cek kedua yaitu x_{k+2} untuk mendeteksi jumlah error genap. Metode ini disebut kode *error-detecting* dan banyak digunakan, seperti pada kode ISBN dan rekening bank.

Bagaimanapun digit cek dapat mendeteksi error, cara tersebut tetap tidak dapat menunjukkan letak error yang terjadi. Artinya, kode *error-detecting* hanya dapat mendeteksi bahwa telah terjadi error, tanpa dapat memperbaikinya. Sebab itu, digunakanlah kode *error-correcting*. Sebelum mengkaji kode *error-correcting* lebih jauh, perlu dibahas model sederhana komunikasi digital seperti ditunjukkan gambar berikut.



Gambar 4.1: Model sederhana komunikasi digital

Bagian (1) pada diagram di atas menunjukkan sumber informasi/pesan. Dalam hal ini pesan dapat dipandang sebagai simbol sepanjang k yaitu $a_1a_2 \dots a_k \in \mathbb{F}_q^k$. Bagian (2) menunjukkan proses encode, yaitu proses transformasi pesan menjadi sebuah kode $\mathbf{x} = x_1x_2 \dots x_n \in \mathbb{F}_q^n$ dengan $n \geq k$. Bagian (3) menunjukkan saluran dengan noise (4) yang dapat mengubah $\mathbf{x}$ menjadi $\mathbf{y} = y_1y_2 \dots y_n \in \mathbb{F}_q^n$ dengan error $\mathbf{e} = \mathbf{y} - \mathbf{x}$. Pada bagian (5) kode $\mathbf{y}$ dikoreksi menjadi $\mathbf{x}$, dan didecode kembali menjadi pesan asli dan akhirnya diterima pada bagian (6).

Definisi 4.1.1. (Lidl dan Pilz, 1998; Hamming, 1986) Sebuah himpunan $C \subseteq \mathbb{F}_q^n$ disebut **kode** atas lapangan $\mathbb{F}_q$ dengan elemen-elemennya disebut *codeword*. Jika $\mathbf{x}$ adalah *codeword* yang ditransmisikan dan $\mathbf{y}$ adalah *codeword* yang diterima, maka *codeword* error adalah $\mathbf{e} = \mathbf{y} - \mathbf{x}$.

4.1.2 Kode linear dan encoder parity check

Pertama akan didefinisikan Jarak Hamming dan Bobot Hamming.

Definisi 4.1.2 (Jarak Hamming)**.** (Hamming, 1986; Lidl dan Pilz, 1998) Jarak Hamming $d(\mathbf{x}, \mathbf{y})$ dari dua vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ dan $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ di $\mathbb{F}_q^n$ adalah banyaknya indeks i di mana $x_i \neq y_i$.

Definisi 4.1.3 (Jarak Hamming)**.** (Hamming, 1986; Lidl dan Pilz, 1998) Bobot Hamming $wt(\mathbf{x})$ dari suatu vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ di $\mathbb{F}_q^n$ adalah banyaknya indeks i di mana $x_i \neq 0$. Dengan kata lain, $wt(\mathbf{x}) = d(\mathbf{x}, \mathbf{0})$.

Teorema 4.1.1. *(Kreyszig, 1978; Hamming, 1986; Lidl dan Pilz, 1998)*

1. *Jika $d(\mathbf{x}, \mathbf{y})$ adalah Jarak Hamming, maka $d(\mathbf{x}, \mathbf{y})$ adalah metrik di $\mathbb{F}_q^n$.*

2. *Jika $wt(\mathbf{x})$ adalah Bobot Hamming, maka $wt(\mathbf{x})$ adalah norma di $\mathbb{F}_2^n$.*

Bukti. 1. Akan dibuktikan bahwa jarak Hamming $d(\mathbf{x}, \mathbf{y})$ adalah metrik di $\mathbb{F}_q^n$. Dari Definisi 2.4.1, syarat 1 terpenuhi karena jarak Hamming $d(\mathbf{x}, \mathbf{y})$ selalu taknegatif dan berhingga. Syarat 2 juga terpenuhi sebab jika $d(\mathbf{x}, \mathbf{y}) = 0$, maka $\mathbf{x} = \mathbf{y}$. Dengan kata lain jika dua vektor memiliki jarak Hamming 0, maka tidak ada elemen-elemennya yang berbeda. Artinya kedua vektor itu sama. Konversnya juga benar, yaitu jika $\mathbf{x} = \mathbf{y}$, maka $d(\mathbf{x}, \mathbf{y})$. Syarat 3 terpenuhi sebab jarak Hamming antara $\mathbf{x}$ dan $\mathbf{y}$ yang dinyatakan dengan $d(\mathbf{x}, \mathbf{y})$ dan jarak Hamming antara $\mathbf{y}$ dan $\mathbf{x}$ yang dinyatakan dengan $d(\mathbf{y}, \mathbf{x})$ pastilah sama. Untuk membuktikan syarat 4 diperlukan fakta bahwa $d(\mathbf{x}, \mathbf{y})$ menyatakan banyaknya

elemen $\mathbf{x}$ yang harus diubah agar menjadi $\mathbf{y}$ dan $0 \leq d(\mathbf{x}, \mathbf{y}) \leq n$. Dari sini dapat dilihat bahwa banyaknya perubahan yang harus dilakukan untuk mengubah $\mathbf{x}$ menjadi $\mathbf{y}$ tidak melebihi jumlah perubahan yang harus dilakukan untuk mengubah $\mathbf{x}$ menjadi $\mathbf{z}$ kemudian mengubah $\mathbf{z}$ menjadi $\mathbf{y}$. Artinya $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$.

2. Akan dibuktikan bahwa Bobot Hamming $wt(\mathbf{x})$ adalah norma di $\mathbb{F}_2^n$. Dari Definisi 2.4.3, syarat 1 terpenuhi karena bobot Hamming dari suatu vektor pastilah taknegatif. Syarat 2 terpenuhi karena jika $wt(\mathbf{x}) = 0$, maka $\mathbf{x} = \mathbf{0}$ sebab semua elemen $\mathbf{x}$ adalah 0. Konversnya juga benar, yaitu jika $\mathbf{x} = \mathbf{0}$, maka $wt(\mathbf{x}) = 0$. Syarat 3 terpenuhi karena untuk sebarang $\alpha \in \mathbb{F}_2$, $wt(\alpha\mathbf{x}) = \alpha wt(\mathbf{x})$. Untuk membuktikan syarat 4, perlu diingat bahwa $0 \leq wt(\mathbf{x}) \leq n$. Jika pada vektor-vektor $\mathbf{x}$ dan $\mathbf{y}$ tidak ada i sehingga $x_i = y_i$, maka berlaku $wt(\mathbf{x} + \mathbf{y}) = wt(\mathbf{x}) + wt(\mathbf{y})$. Jika pada vektor-vektor $\mathbf{x}$ dan $\mathbf{y}$ terdapat beberapa i sehingga $x_i = y_i$, maka $wt(\mathbf{x} + \mathbf{y}) < wt(\mathbf{x}) + wt(\mathbf{y})$ (karena $\mathbf{x}$ dan $\mathbf{y}$ vektor biner). Artinya berlaku $wt(\mathbf{x} + \mathbf{y}) \leq wt(\mathbf{x}) + wt(\mathbf{y})$ sehingga syarat 4 terpenuhi.

□

Definisi 4.1.4. (Lidl dan Pilz, 1998) Jarak minimum dari suatu kode C adalah

$$d_{min}(C) = \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}$$

Sebelum mendefinisikan kode linear, perlu ditinjau contoh berikut.

Contoh 4.1.1. Diberikan pesan biner $x_1x_2x_3$ dan digit cek

$x_4x_5x_6$ dengan

$$x_4 = x_1 + x_2$$
$$x_5 = x_1 + x_3$$
$$x_6 = x_2 + x_3$$

atau

$$x_1 + x_2 + x_4 = 0$$
$$x_1 + x_3 + x_5 = 0$$
$$x_2 + x_3 + x_6 = 0$$

Sistem persamaan ini digunakan untuk memperoleh *codeword*. Misal diberikan pesan 010, maka pesan ini dapat dikodekan menjadi 010101. Semua *codeword* dapat diperoleh dengan cara yang sama dan terdapat sebanyak 8 *codeword* dalam kode ini.

Sistem persamaan di atas dapat dinyatakan sebagai $H\mathbf{x} = \mathbf{0}$ dengan $\mathbf{x}$ solusi dari sistem persamaan tersebut dan

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Ingat bahwa himpunan semua $\mathbf{x}$ yang memenuhi persamaan di atas adalah kernel dari H atau nol(H). ■

Matriks H pada contoh di atas disebut **matriks cek** berukuran $(n-k) \times n$ atas $\mathbb{F}_q$, dengan k panjang pesan dan $n-k$ panjang digit cek. Matriks H berbentuk $(A \mid I_{n-k})$ dengan A adalah matriks berukuran $(n-k) \times k$ dan I_{n-k} adalah matriks identitas berukuran $n-k$. Pada Contoh 4.1.1 di atas, $k = 3$, $n = 3$, dan $H = (A \mid I_3)$. Dapat diperhatikan bahwa rank(H) pastilah $n-k$, sebab adanya matriks I_{n-k} menjamin baris-baris H selalu bebas linear.

Definisi 4.1.5 (Kode Linear)**.** (Lidl dan Pilz, 1998) Diberikan matriks cek H berukuran $(n-k)\times n$ atas $\mathbb{F}_q$ dengan rank $n-k$. Himpunan

$$C = \{\mathbf{x} : H\mathbf{x}^T = \mathbf{0}, \mathbf{x} \in \mathbb{F}_q^n\}$$

disebut **kode linear**. Jika k panjang pesan, $n-k$ panjang digit cek, dan $d_{min}(C) = d$, maka C disebut kode linear (n, k, d). Laju informasi dinyatakan sebagai k/n. Jika $q = 2$, maka C disebut kode biner.

Penggunaan matriks cek dalam membentuk *codeword* adalah salah satu metode encoding dalam kode linear yang disebut encoder *Parity Check*. Dengan kata lain matriks H memetakan elemen-elemen himpunan pesan ke elemen-elemen himpunan *codeword*, atau $H : \mathbb{F}_q^k \to \mathbb{F}_q^n$. Selanjutnya akan didefinisikan metode decoding yang disebut decoder *Nearest Neighbor Decoding* menggunakan Definisi 2.4.2.

4.1.3 Decoder nearest neighbor

Misal sebuah pesan diencode menjadi *codeword* $\mathbf{x} \in C$ lalu ditransmisikan dan terdapat error sehingga berubah menjadi $\mathbf{y} \in \mathbb{F}_q^n$. Penerima dapat menentukan $\mathbf{x}$ sebagai vektor yang paling dekat dengan $\mathbf{y}$ (dalam hal jarak Hamming).

Definisi 4.1.6 (Bola)**.** (Lidl dan Pilz, 1998) Diberikan kode linear $C \subseteq \mathbb{F}_q^n$. Himpunan $S_r(\mathbf{x}) = \{\mathbf{y} \in \mathbb{F}_q^n : d(\mathbf{x}, \mathbf{y}) \leq r, \mathbf{x} \in C\}$ disebut bola dengan jari-jari r.

Jika *codeword* $\mathbf{y}$ diterima, maka $\mathbf{y}$ didecode menjadi *codeword* $\mathbf{x}$ di mana $\mathbf{y} \in S_r(\mathbf{x})$. Misal, pada Contoh 4.1.1:

$$S_1(010101) = \{010101, 110101, 000101, 011101,$$
$$010001, 010111, 010100\}$$

sehingga jika 000101 diterima, maka *codeword* ini akan didecode menjadi 010101.

Tetapi untuk beberapa nilai r tertentu, mungkin saja suatu vektor termuat dalam lebih dari satu bola sehingga $\mathbf{y}$ tidak dapat didecode ke salah satu *codeword*. Artinya bola-bola ini tidak boleh saling beririsan, atau $r < \lfloor \frac{1}{2} d_{min}(C) \rfloor$.

Teorema 4.1.2. *(Lidl dan Pilz, 1998) Jika C adalah kode linear dengan $d_{min}(C) = d$, maka C dapat mendeteksi hingga $d-1$ error dan dapat mengkoreksi hingga $\left\lfloor \frac{d-1}{2} \right\rfloor$ error.*

Bukti. Pertama akan dibuktikan bahwa kode linear C dengan $d_{min}(C) = d$ dapat mendeteksi hingga $d-1$ error. Misal *codeword* $\mathbf{x} \in C$ ditransmisikan dan vektor $\mathbf{y} \in \mathbb{F}_q^n$ diterima dengan jumlah error $\leq d-1$. Karena jarak minimum di C adalah d, maka $\mathbf{y}$ tidak mungkin merupakan suatu *codeword* lain di C, sehingga error dapat dideteksi.

Kedua, kan dibuktikan bahwa C dapat mengkoreksi hingga $\left\lfloor \frac{d-1}{2} \right\rfloor$ error. Misal *codeword* $\mathbf{x} \in C$ ditransmisikan dan vektor $\mathbf{y} \in \mathbb{F}_q^n$ diterima dengan jumlah error $\leq \left\lfloor \frac{d-1}{2} \right\rfloor$. Karena jari-jari dari setiap bola S pasti kurang dari $\left\lfloor \frac{d}{2} \right\rfloor$ dan $\left\lfloor \frac{d-1}{2} \right\rfloor < \left\lfloor \frac{d}{2} \right\rfloor$ untuk setiap d, pastilah $\mathbf{y}$ termuat dalam bola dengan pusat $\mathbf{x}$, yaitu $\mathbf{y} \in S_r(\mathbf{x})$. Artinya $\mathbf{y}$ selalu dapat dikoreksi dengan benar menjadi $\mathbf{x}$ untuk error $\leq \left\lfloor \frac{d-1}{2} \right\rfloor$. □

Teorema 4.1.3 (Batasan Hamming)**.** *(Lidl dan Pilz, 1998; Hamming, 1986) Jika $C \subseteq \mathbb{F}_q^n$ adalah kode linear sepanjang n, dengan $|C| = M$, $d_{min}(C) = d$, dan dapat mengkoreksi hingga $t = \lfloor \frac{d-1}{2} \rfloor$ error, maka kode C memenuhi pertidaksamaan:*

$$M\left(1 + (q-1)\binom{n}{1} + \ldots (q-1)^t \binom{n}{t}\right) \leq q^n$$

Bukti. Pertama, banyaknya semua vektor dalam $\mathbb{F}_q^n$ adalah q^n. Dalam $\mathbb{F}_q$ terdapat $(q-1)^r \binom{n}{r}$ vektor sepanjang n dan berbobot r. Hal ini dikarenakan terdapat $(q-1)^r$ pilihan digit (elemen $\mathbb{F}_q$) untuk mengubah sebanyak r elemen $\mathbf{x}$ dan

karena terdapat $\binom{n}{r}$ cara untuk memilih elemen $\mathbf{x}$ mana saja yang akan diubah. Karena $t = \lfloor \frac{d-1}{2} \rfloor$ maka setiap bola S_t pasti saling asing. Dalam setiap bola ini terdapat sebanyak $1+(q-1)\binom{n}{1}+\ldots+(q-1)^t\binom{n}{t}$ vektor. Jika banyaknya elemen C adalah M maka terdapat $M\left(1+(q-1)\binom{n}{1}+\ldots(q-1)^t\binom{n}{t}\right)$ vektor yang termuat dalam bola-bola tersebut. Karena bola-bola ini tidak saling beririsan, maka pastilah jumlah seluruh vektor yang termuat dalam bola tidak melebihi jumlah keseluruhan vektor yang ada di $\mathbb{F}_q^n$, yaitu q^n. □

Kode C yang memenuhi kesamaan pada Teorema 4.1.3 disebut kode sempurna. Artinya setiap elemen $\mathbb{F}_q^n$ pasti termuat di $S_r(\mathbf{x})$ dengan $\mathbf{x} \in C$.

4.1.4 Encoder matriks generator

Pada bagian ini didefinisikan metode encoding lainnya yang disebut encoder Matriks Generator. Dari definisi matriks cek H diketahui bahwa $H\mathbf{x}^T = \mathbf{0}$. Diberikan pesan $\mathbf{a} = a_1a_2\ldots a_k$ dan *codeword* $\mathbf{x} = x_1x_2\ldots x_kx_{k+1}\ldots x_n$. Misal $\mathbf{x}_k = x_1\ldots x_k$ dan $\mathbf{x}_n = x_{k+1}\ldots x_n$. Dari sini diperoleh:

$$H\mathbf{x}^T = \begin{pmatrix} A & I_{n-k} \end{pmatrix} \begin{pmatrix} \mathbf{x}_k^T \\ \\ \mathbf{x}_n^T \end{pmatrix}$$

$$\mathbf{0} = A\mathbf{x}_k^T + I_{n-k}\,\mathbf{x}_n^T$$

$$\mathbf{x}_n^T = -A\mathbf{x}_k^T$$

Karena $\mathbf{x}_k^T = \mathbf{a}^T$, maka $\mathbf{x}_n^T = -A\mathbf{a}^T$. Artinya

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} I_k \\ -A \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix}$$

$$\mathbf{x} = \mathbf{a}\left(I_k \mid -A^T\right)$$

Definisi 4.1.7 (Matriks Generator)**.** (Lidl dan Pilz, 1998) Diberikan C kode linear dengan matriks cek $H = (A \mid I_{n-k})$. Matriks $G = (I_k \mid -A^{\mathrm{T}})$ disebut **matriks generator** dari C. *Codeword* x bisa diperoleh dengan $\mathbf{a}G = \mathbf{x}$ dengan $\mathbf{a}$ adalah pesan sepanjang k.

Dari definisi jelas bahwa $GH^{\mathrm{T}} = \mathbf{0}$ sebab

$$\begin{aligned} GH^{\mathrm{T}} &= (I_k \mid -A^{\mathrm{T}}) \begin{pmatrix} A^{\mathrm{T}} \\ I_{n-k} \end{pmatrix} \\ &= (I_k A^{\mathrm{T}}) - (A^{\mathrm{T}} I_{n-k}) \\ &= (\mathbf{0})_{k \times n} \end{aligned}$$

Teorema 4.1.4. *(Lidl dan Pilz, 1998) Jika G adalah matriks generator dari kode linear C, maka baris-baris G membentuk basis dari C.*

Bukti. Menurut definisi G, k baris dari G jelas saling bebas linear. Jika $\mathbf{u}$ adalah vektor baris dari G, maka menurut Definisi 4.1.5 $\mathbf{u}H^{\mathrm{T}} = \mathbf{0}$ atau $H\mathbf{u}^{T} = \mathbf{0}$. Artinya $\mathbf{u} \in C$. Perhatikan bahwa $\dim(C)$ tak lain adalah $\dim(\mathrm{nol}(H)) = n - \mathrm{rank}(H) = k$. Karena vektor-vektor baris G sebanyak k, sepanjang n dan saling bebas linear, maka pastilah vektor-vektor baris G membentang C. □

Berikut adalah pembahasan mengenai suatu metode lain dalam pencarian jarak minimum di C yang lebih cepat secara komputasi.

Teorema 4.1.5. *(Lidl dan Pilz, 1998) Jika C adalah kode linear (n, k, d), maka $d = min\{wt(\mathbf{x}) : \mathbf{x} \in C\}$.*

Bukti. Untuk membuktikan teorema ini perlu diketahui bahwa $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x} - \mathbf{y}, \mathbf{0})$. Dari Definisi 4.1.4 diketahui bahwa

$$\begin{aligned} d_{min}(C) &= \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\} \\ &= \min\{d(\mathbf{x} - \mathbf{y}, \mathbf{0}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\} \\ &= \min\{wt(\mathbf{x} - \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\} \end{aligned}$$

□

Teorema ini jelas mempercepat penghitungan jarak minimum sebab hanya membutuhkan sebanyak $M-1$ penghitungan daripada cara sebelumnya yang membutuhkan $\binom{M}{2}$ kali penghitungan dengan M banyaknya *codeword* di kode C. Misal $\text{mld}(H)$ menyatakan banyaknya kolom minimal yang bergantungan linear di matriks H. Maka teorema berikut ini berlaku.

Teorema 4.1.6. *(Lidl dan Pilz, 1998) Jika H adalah matriks cek untuk suatu C kode linear (n, k, d), maka berlaku:*

1. *$dim(C) = k = n - rank(H)$,*
2. *$d = mld(H)$,*
3. *$d \leq n - k + 1$.*

Bukti. 1. Cukup jelas sebab $\text{rank}(H) = n-k$, $n - \text{rank}(H) = k = \dim(C)$. 2. Misal H memiliki kolom-kolom $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$. Jika $\mathbf{x} = x_1 x_2 \ldots x_n \in C$ maka $H\mathbf{x}^T = x_1\mathbf{u}_1 + x_2\mathbf{u}_2 + \ldots + x_n\mathbf{u}_n = \mathbf{0}$. Jika $\mathbf{x}$ adalah vektor dengan bobot minimum maka pasti terdapat sebanyak d elemen tak nol di $\mathbf{x}$, atau terdapat sebanyak d kolom H yang saling bergantungan linear. 3. Karena $\text{mld}(H) \leq \text{rank}(H) + 1$ dan dari 2., maka

$$\begin{aligned} d &= \text{mld}(H) \\ &\leq \text{rank}(H) + 1 \\ &\leq n - k + 1 \end{aligned}$$

□

Contoh 4.1.2. Melanjutkan kembali Contoh 4.1.1, dapat dibentuk kode linear C dengan matriks cek

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

panjang $k = 3$, $n - k = 3$, laju informasi $k/n = 1/2$, jarak minimum $d = 3$, kemampuan mendeteksi error $d - 1 = 2$, kemampuan mengkoreksi error $\lfloor \frac{d-1}{2} \rfloor = 1$. Misal diberikan pesan 100 dan diencode menjadi 100110. *Codeword* ini ditransmisikan dan diberi satu error sehingga menjadi 101110. Untuk setiap $\mathbf{x} \in C$, terdapat bola $S_1(\mathbf{x})$ dengan jari-jari 1. Artinya jika 101110 diterima, *codeword* ini akan dikoreksi menjadi 100110. Tetapi jika *codeword* 111110 diterima, maka *codeword* akan dikoreksi menjadi 011110. Terlihat bahwa C tidak mampu mengkoreksi error yang lebih dari 1.

Matriks generator dari C adalah

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

di mana untuk semua pesan $\mathbf{a} = a_1a_2a_3 \in \mathbb{F}_2^3$, C dapat diperoleh dengan $C = \{\mathbf{x} : \mathbf{x} = \mathbf{a}G\}$. ■

Teorema 4.1.7 (Batasan Plotkin)**.** *(Lidl dan Pilz, 1998) Jika C kode linear (n, k, d) atas $\mathbb{F}_q$ yang memuat sebanyak M codeword, maka berlaku*

$$d \leq \frac{nM(q-1)}{(M-1)q}$$

Bukti. Misal C kode linear (n, k) atas $\mathbb{F}_q$ dan misal $1 \leq i \leq n$ sehingga C memuat sebuah *codeword* yang elemen ke-i nya taknol. Misal D adalah subruang dari C yang memuat semua *codeword* yang elemen ke-i nya nol. Di C/D terdapat q elemen, yang menyatakan banyaknya pilihan elemen ke-i. Jika $|C| = M = q^k$ adalah banyaknya elemen di C, maka $M/|D| = |C/D|$, atau dengan kata lain $|D| = q^{k-1}$. Jumlahan dari semua bobot elemen C adalah $\leq nq^{k-1}(q-1)$. Karena

jumlah total *codeword* yang bobotnya tak nol adalah $q^k - 1$, maka $nq^{k-1} \geq$ jumlah semua bobot $\geq d(q^k - 1)$. Artinya

$$d \leq \frac{nq^{k-1}(q-1)}{q^k - 1} \leq \frac{nq^k(q-1)}{(q^k-1)q} = \frac{nM(q-1)}{(M-1)q}$$

□

Misal C kode linear (n, k) atas $\mathbb{F}_q$ dengan $d_{min}(C) = d$. Artinya C memiliki radius koreksi hingga $\left\lfloor \frac{d-1}{2} \right\rfloor$. Untuk memperbesar radius koreksi, maka jarak minimum d harus diperbesar hingga d', dengan $d' > d$. Dengan ini dapat dikonstruksi C' yang merupakan kode linear (n, k), dengan radius koreksi $\left\lfloor \frac{d'-1}{2} \right\rfloor$. Karena jarak minimum diperbesar, maka jumlah *codeword* yang dapat digunakan semakin sedikit. Hal ini dikarenakan saat jari-jari bola $S_r(\mathbf{x})$ membesar, bola-bola tersebut tetap tidak boleh beririsan. Akibatnya, *codeword* yang termuat dalam bola-bola tersebut semakin sedikit. Namun, eksistensi kode linear yang akan dikonstruksi tersebut harus diperiksa menggunakan batasan Hamming dan batasan Plotkin sebelum dikonstruksi.

4.1.5 Decoder syndrome

Pada bagian ini akan dibahas suatu metode decoding lain untuk kode linear yang disebut decoder Syndrome.

Teorema 4.1.8. *(Lidl dan Pilz, 1998; Neubauer dkk., 2007) Jika codeword* $\mathbf{y}$ *diterima, maka errornya adalah vektor* $\mathbf{e}$ *dengan bobot minimum yang berada di koset yang juga memuat* $\mathbf{y}$*. Jika* $\mathbf{e}$ *adalah error, maka* $\mathbf{y}$ *dikoreksi menjadi* $\mathbf{x}$ *yaitu* $\mathbf{x} = \mathbf{y} - \mathbf{e}$*.*

Bukti. Pandang C sebagai subruang dari $\mathbb{F}_q^n$. Himpunan $\mathbb{F}_q^n/C$ memuat semua koset $\mathbf{a} + C = \{\mathbf{a} + \mathbf{x} : \mathbf{x} \in C\}$ untuk sebarang $\mathbf{a} \in \mathbb{F}_q^n$. Setiap koset memuat $|C| = q^k$ vektor. Artinya

terdapat partisi

$$\mathbb{F}_q^n = C \cup (\mathbf{a}^{(1)} + C) \cup (\mathbf{a}^{(2)} + C) \cup \ldots \cup (\mathbf{a}^{(t)} + C)$$

dengan $t = q^{n-k} - 1$. Jika sebuah vektor $\mathbf{y}$ diterima, pastilah $\mathbf{y}$ termuat dalam salah satu koset tersebut, sebut saja koset $\mathbf{a}^{(i)} + C$ untuk suatu i. Jika *codeword* $\mathbf{x}^{(1)}$ ditransmisikan, maka error $\mathbf{e}$ pasti juga berada di koset $\mathbf{a}^{(i)} + C$, yaitu

$$\mathbf{e} = \mathbf{y} - \mathbf{x}^{(1)} \in \mathbf{a}^{(i)} + C - \mathbf{x}^{(1)} = \mathbf{a}^{(i)} + C$$

□

Definisi 4.1.8 (Pimpinan Koset)**.** Vektor dengan bobot minimum dalam suatu koset disebut **pimpinan koset**.

Misal $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(t)}$ pimpinan koset, maka dapat dikonstruksi tabel koset (Lidl dan Pilz, 1998):

$$\begin{array}{cccc}
\mathbf{x}^{(1)} = \mathbf{0} & \mathbf{x}^{(2)} & \ldots & \mathbf{x}^{(q^k)} \\
\mathbf{a}^{(1)} + \mathbf{x}^{(1)} & \mathbf{a}^{(1)} + \mathbf{x}^{(2)} & \ldots & \mathbf{a}^{(1)} + \mathbf{x}^{(q^k)} \\
\ldots & \ldots & \ldots & \ldots \\
\mathbf{a}^{(t)} + \mathbf{x}^{(1)} & \mathbf{a}^{(t)} + \mathbf{x}^{(2)} & \ldots & \mathbf{a}^{(t)} + \mathbf{x}^{(q^k)}
\end{array}$$

dengan baris pertama menyatakan elemen-elemen C dan kolom pertama menyatakan pimpinan-pimpinan koset. Jika sebuah vektor $\mathbf{y}$ diterima, maka $\mathbf{y}$ harus dicari di tabel. Misal $\mathbf{y} = \mathbf{a}^{(i)} + \mathbf{x}^{(j)}$. Maka error $\mathbf{e} = \mathbf{a}^{(i)}$ karena $\mathbf{a}^{(i)}$ memiliki bobot minimum. Vektor $\mathbf{y}$ akan dikoreksi menjadi $\mathbf{x} = \mathbf{y} - \mathbf{e} = \mathbf{y} - \mathbf{a}^{(i)} = \mathbf{x}^{(j)}$ yaitu elemen pertama pada kolom di mana $\mathbf{y}$ ditemukan.

Metode decoding ini cukup mudah dilakukan. Tetapi untuk ukuran kode yang lebih besar, pencarian $\mathbf{y}$ di tabel berukuran $(q^{n-k} - 1) \times q^k$ bisa membutuhkan waktu yang

sangat lama. Misal jika diambil kode pada Contoh 4.1.1, tabel kosetnya berukuran 7×8. Semakin besar q, n dan k, ukuran tabel juga membesar secara signifikan. Hal ini juga berpengaruh pada penggunaan memori yang lebih besar untuk menyimpan tabel koset berukuran besar. Sebab itu, perlu dikembangkan metode decoding yang lebih cepat.

Perlu diperhatikan bahwa untuk setiap vektor $\mathbf{v}$ dalam satu koset (dalam satu baris pada tabel), nilai $H\mathbf{v}^T$ adalah sama, sebab $H\mathbf{v}^T = H(\mathbf{v}^T + \mathbf{a}^T) = H\mathbf{v}^T + H\mathbf{a}^T = H\mathbf{a}^T$ dengan $\mathbf{a}$ pimpinan koset. Dari sini dapat didefinisikan syndrome.

Definisi 4.1.9 (Syndrome)**.** (Lidl dan Pilz, 1998) Misal H adalah matriks cek dari kode linear (n, k). Vektor $S(\mathbf{y}) = H\mathbf{y}^T$ dengan panjang $n - k$ disebut **syndrome** dari $\mathbf{y}$.

Teorema 4.1.9. *(Lidl dan Pilz, 1998; Neubauer dkk., 2007) Jika C kode linear (n, k) dan $S(\mathbf{y})$ syndrome dari vektor $\mathbf{y} \in \mathbb{F}_q^n$, maka berlaku:*

1. $S(\mathbf{y}) = \mathbf{0} \iff \mathbf{y} \in C$,
2. $S(\mathbf{y}) = S(\mathbf{z}) \iff \mathbf{y} + C = \mathbf{z} + C \iff \mathbf{y}$ *dan* $\mathbf{z}$ *berada di koset yang sama.*

Bukti. 1. Jika $S(\mathbf{y}) = \mathbf{0}$, jelas bahwa $\mathbf{y} \in C$. Konversnya, jika $\mathbf{y} \in C$ maka $S(\mathbf{y}) = \mathbf{0}$ (sesuai dengan Definisi 4.1.5),

2. Misal $S(\mathbf{y}) = S(\mathbf{z})$

$$\begin{aligned} S(\mathbf{y}) = S(\mathbf{z}) &\iff H\mathbf{y}^T = H\mathbf{z}^T \\ &\iff H(\mathbf{y}^T - \mathbf{z}^T) = \mathbf{0} \\ &\iff \mathbf{y} - \mathbf{z} \in C \\ &\iff \mathbf{y} + C = \mathbf{z} + C \end{aligned}$$

Artinya $\mathbf{y}$ dan $\mathbf{z}$ berada dalam koset yang sama. □

Penentuan pimpinan koset dengan pencarian langsung dalam tabel koset membutuhkan waktu yang cukup lama. Dengan memanfaatkan sifat syndrome, akan dibahas decoder syndrome yang lebih cepat secara komputasi daripada metode pada Teorema 4.1.8.

Teorema 4.1.10 (Decoder Syndrome)**.** *(Lidl dan Pilz, 1998; Neubauer dkk., 2007) Misal C kode linear (n, k, d) dan misal codeword $\mathbf{x} \in C$ ditransmisikan. Jika vektor $\mathbf{y} \in \mathbb{F}_q^n$ diterima, maka dihitung $S(\mathbf{y}) = S(\mathbf{e})$. Jika error adalah vektor $\mathbf{e}$, maka vektor $\mathbf{y}$ didecode menjadi $\mathbf{x}$, yaitu $\mathbf{x} = \mathbf{y} - \mathbf{e}$.*

Bukti. Sekarang koset dapat ditentukan dengan menggunakan syndrome. Misal $\mathbf{e} = \mathbf{y} - \mathbf{x}$, $\mathbf{x} \in C$, dan $\mathbf{y} \in \mathbb{F}_q^n$. Selanjutnya dihitung $S(\mathbf{y}) = S(\mathbf{x} + \mathbf{e}) = S(\mathbf{x}) + S(\mathbf{e}) = S(\mathbf{e})$ ($\mathbf{y}$ dan $\mathbf{e}$ ada dalam satu koset). Vektor $\mathbf{e}$ adalah vektor dengan bobot terkecil, yaitu pimpinan koset. Setelah diperoleh $\mathbf{e}$, vektor $\mathbf{y}$ dikoreksi menjadi *codeword* yang paling memungkinkan untuk benar, yaitu $\mathbf{x} = \mathbf{y} - \mathbf{e}$. □

Contoh 4.1.3. Melanjutkan Contoh 4.1.2, maka dapat diketahui bahwa terdapat $q^{n-k} = 8$ pimpinan koset di C. Dengan menggunakan H, dapat dikonstruksi tabel syndrome-pimpinan koset.

Jika $\mathbf{y} = 101001$ diterima, dihitung $S(\mathbf{y}) = H\mathbf{y}^T = 100$. Dari tabel dapat dilihat bahwa pimpinan koset dengan syndrome 100 adalah 000100. Artinya $\mathbf{y}$ dapat dikoreksi menjadi $\mathbf{x} = 101001 - 000100 = 101101$. ■

Decoder syndrome ini lebih baik daripada decoder nearest neighbor. Namun untuk ukuran kode yang cukup besar, metode ini masih membutuhkan waktu yang cukup lama. Misal untuk kode $(50, 20)$ atas $\mathbb{F}_2$, banyaknya pimpinan koset adalah 2^{30}. Khusus untuk kasus kode biner dengan error maksimum 1, ada cara untuk mempercepat decoder syndrome

Tabel 4.1: Tabel Syndrome Contoh 4.1.2

pimpinan	syndrome
000000	000
100000	011
010000	101
001000	110
000100	100
000010	010
000001	001
001001	111

sebagai berikut. Misal diberikan C kode linear (n, k, d) atas $\mathbb{F}_2$ dengan matriks cek H dan misalkan vektor $\mathbf{y}$ diterima. Jika syndrome $S(\mathbf{y})$ sama dengan kolom ke-j dari H, maka error terjadi pada elemen ke-j dari $\mathbf{y}$ (Hamming, 1986; Lidl dan Pilz, 1998). Cara ini memungkinkan koreksi satu error tanpa menggunakan tabel syndrome.

Dengan demikian telah dibahas dua metode encoding yaitu encoder matriks cek, matriks generator, dan dua metode decoding yaitu decoder nearest neighbor dan syndrome. Berikut akan dibahas salah satu bentuk khusus kode linear yaitu kode Hamming.

4.2 Kode Hamming

Pertama akan didefinisikan kode Hamming atas $\mathbb{F}_2$.

Definisi 4.2.1 (Kode Hamming)**.** (Hamming, 1986; Lidl dan Pilz, 1998) Sebuah kode biner C_m sepanjang $n = 2^m - 1, m \geq 2$ dengan matriks cek H berukuran $m \times (2^m - 1)$ yang semua kolom-kolomnya taknol disebut kode Hamming.

Sebarang dua kolom di H saling bebas linear. Namun, tidak sebarang tiga kolom di H saling bebas linear. Artinya

$\text{mld}(H) = 3$. Artinya kode Hamming adalah kode linear $(2^m - 1, 2^m - 1 - m, 3)$, dapat mendeteksi error ≤ 2 dan dapat mengkoreksi 1 error.

Untuk kode Hamming biner, dapat digunakan metode khusus dalam penentuan error dan koreksi. Misal kolom-kolom H diurutkan sedemikian hingga kolom ke-i adalah representasi biner dari i. Jika $\mathbf{y}$ diterima dan $S(\mathbf{y}) = H\mathbf{y}^T = H\mathbf{e}^T$ adalah kolom ke-i dari H, maka error terjadi pada kolom ke-i pada $\mathbf{y}$.

Contoh 4.2.1. Diberikan C_3 kode Hamming $(7, 4, 3)$ dengan matriks cek

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Dapat dilihat bahwa kolom ke-i dari H adalah representasi biner dari i. Jika $S(\mathbf{y}) = 101$, maka error terjadi pada elemen ke-5, sebab 101 adalah kolom ke-5 dari H. ■

Kode Hamming C_m adalah kode sempurna sebab memenuhi kesamaan Teorema 4.1.3.

4.3 Perancangan Simulasi

Pada subbab ini akan dirancang proses simulasi transmisi data sesuai dengan diagram yang ditunjukkan pada Gambar 3, yang melibatkan representasi data, encoding, transmisi, koreksi, dan decoding. Tipe data yang digunakan adalah teks dan gambar. Kode yang akan digunakan adalah kode linear dan kode Hamming atas $\mathbb{F}_2$.

4.3.1 Pendefinisian data dan representasinya

Data yang akan disimulasikan adalah teks dan gambar. Teks dalam hal ini himpunan berhingga yang dibentuk dari elemen-elemen himpunan alfanumerik, spasi, dan titik (.)

yaitu $S = \{$, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, .$\}$ yang terdiri dari 64 elemen. String $u_i \in T$ dapat direpresentasikan dalam biner $\geq$ 6-bit dengan menggunakan representasi biner indeksnya, yaitu i. Misal string 'c' berada pada indeks 3, artinya representasi 'c' adalah 000011. Data ini akan disimpan dalam bentuk **dictionary**, yaitu bentuk data khusus dalam Python yang berupa list (array dalam beberapa bahasa pemrograman lain) dengan bentuk $\{key_1 : value_1, \ldots, key_n : value_n\}$. Kelebihan dictionary adalah kemudahan dalam memanggil $value_i$ dengan cara memanggil key_i yang berkaitan. Dengan looping, akan dibuat sebuah dictionary pertama dengan bentuk $\{string : biner\}$ yang akan digunakan dalam proses encoding. Selanjutnya akan dibuat dictionary lainnya dengan bentuk $\{biner : string\}$ (kebalikan dari dictionary pertama) untuk proses koreksi dan decoding.

Gambar (citra digital) yang akan digunakan adalah gambar grayscale. Gambar dalam Python adalah list 2 dimensi yang elemen-elemennya adalah integer $0 \ldots 255$. Artinya integer-integer ini dapat direpresentasikan dalam biner $\geq$ 8-bit. Dibuat dua dictionary untuk proses encoding, koreksi dan decoding masing-masing dengan bentuk $\{integer : biner\}$ dan $\{biner : integer\}$. Untuk memanipulasi gambar digunakan modul Python yaitu OpenCV (http://opencv.org/).

4.3.2 Konstruksi kode

Untuk program simulasi pertama dengan data teks, digunakan kode linear $(12, 6, 3)$ dengan matriks cek

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

dan matriks generator

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Kode ini memiliki jarak minimum $d = 3$, dapat mendeteksi 2 error, mengkoreksi 1 error dengan laju informasi $1/2$. Kode C memuat 64 *codeword*. Kode ini memenuhi Teorema 4.1.3 (Batasan Hamming), yaitu

$$64 \left(1 + \binom{12}{1}\right) = 64(13) = 832 < 2^{12}$$

Kode ini juga memenuhi Teorema 4.1.7 (Batasan Plotkin), yaitu

$$\begin{aligned} 3 &\leq \frac{12 \cdot 64}{2 \cdot 63} \\ &\leq 6,095 \end{aligned}$$

Pada program simulasi kedua dengan data gambar, digunakan kode linear $(16, 8, 3)$ dengan matriks cek

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

dan matriks generator

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Kode ini memiliki jarak minimum $d = 3$, dapat mendeteksi 2 error, mengkoreksi 1 error dengan laju informasi 1/2.Kode C memuat 256 *codeword*. Kode ini memenuhi Teorema 4.1.3 (Batasan Hamming), yaitu

$$256\left(1 + \binom{16}{1}\right) = 256(17) = 4352 < 2^{16}$$

Kode ini juga memenuhi Teorema 4.1.7 (Batasan Plotkin), yaitu

$$\begin{aligned} 3 &\leq \frac{16 \cdot 256}{2 \cdot 255} \\ &\leq 8,031 \end{aligned}$$

Program simulasi ketiga dengan data teks, digunakan kode Hamming dengan $m = 4$ yaitu kode Hamming $(15, 11, 3)$ dengan matriks cek

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Kode ini memiliki jarak minimum $d = 3$, dapat mendeteksi 2 error, mengkoreksi 1 error dengan laju informasi 11/15. Kode C memuat 2048 *codeword*. Kode ini adalah kode sempurna sebab memenuhi kesamaan Teorema 4.1.3 (Batasan Hamming), yaitu

$$2048 \left(1 + \binom{15}{1}\right) = 2048(16) = 32768 = 2^{15}$$

Kode ini juga memenuhi Teorema 4.1.7 (Batasan Plotkin), yaitu

$$\begin{aligned} 3 &\leq \frac{15 \cdot 2048}{2 \cdot 2047} \\ &\leq 7,503 \end{aligned}$$

Program simulasi keempat dengan data gambar, menggunakan kode yang sama dengan yang digunakan pada program simulasi ketiga, yaitu kode Hamming $(15, 11, 3)$.

4.3.3 Alur program simulasi

Misal diberikan kode C dengan panjang n, pesan sepanjang k disebut *input*, *codeword* di C yang disebut *codeword*, tabel data-biner *dict1*, tabel biner-data *dict2*, saluran bernama *channel* dengan error bernama *error*, dan suatu metode encoder dan decoder tertentu. Secara umum proses simulasi dapat dituliskan dalam pseudocode sebagai berikut:

Require: $C, input, dict1, dict2, error$
1: **procedure** SIMULASI($C, input, encoder, decoder, error$)
2: $hasil \leftarrow [\,]$ ▷ List kosong 5 kolom tipe string
3: **for** $i \leftarrow 0, k-1$ **do**
4: $codeword \leftarrow dict1[input[i]]$
5: $encoded \leftarrow C.\textbf{encode}(codeword)$
6: $encoded.\textbf{append}(hasil[0])$
7: $trans \leftarrow channel.\textbf{transmit}(encoded, error)$
8: $trans.\textbf{append}(hasil[1])$
9: $corrected \leftarrow C.\textbf{correct}(transmitted)$
10: $corrected.\textbf{append}(hasil[2])$
11: $decoded \leftarrow C.\textbf{decode}(corrected)$
12: $decoded.\textbf{append}(hasil[3])$
13: $msg \leftarrow dict2[decoded]$
14: $msg.\textbf{append}(hasil[4])$
15: **end for**
16: **for all** $j \in hasil$ **do**
17: **print** j ▷ Menampilkan output simulasi
18: **end for**
19: **end procedure**

4.3.4 Struktur tampilan program simulasi

Program simulasi akan dibuat dengan modul Sage yang disebut interact. Keempat program simulasi memiliki struktur tampilan dasar sebagai berikut.

1. Judul
   Menyatakan judul simulasi dan kode yang digunakan,

2. Input data
   Menampilkan input data yang akan diproses. Pada kasus teks menampilkan kotak input teks, pada kasus gambar menampilkan menu dropdown untuk memilih gambar yang akan diproses,

3. Encoder
   Menampilkan menu dropdown untuk memilih encoder,

4. Decoder
   Menampilkan menu dropdown untuk memilih decoder,

5. Error
   Menampilkan slider untuk menentukan jumlah error yang akan diberikan pada saluran transmisi pada rentang $0 \ldots n$ dengan n panjang *codeword*,

6. Hasil simulasi
   Menampilkan hasil simulasi yang terdiri dari waktu komputasi, kode, matriks cek, matriks generator, jarak minimum, data awal, laju informasi, data dalam biner, data setelah diencode, data setelah ditransmisikan (dengan error bila ada), error yang terjadi, hasil koreksi, hasil decode, dan data yang diterima.

4.4 Implementasi Simulasi Menggunakan Sage

Pendefinisian data teks dan representasi binernya dapat dilakukan di Sage dengan:

```
1  ####################################
2  # dict1 = representasi biner data  #
3  # dict2 = Kebalikan dict1          #
4  ####################################
5
6  strings = ' abcdefghijklmnopqrstuvwxyzABCDEFGHI
       JKLMNOPQRSTUVWXYZ1234567890.'
7  dict1 = {}
8  for i in xrange(len(strings)):
9      dict1[str(strings[i])] = '{0:06b}'.format(i)
10
11 key_dict1 = dict1.keys()
12 val_dict1 = dict1.values()
13
14 dict2 = {}
```

```
for j in xrange(len(strings)):
    dict2[str(val_dict1[j])] = key_dict1[j]
```

Source Code 4.4.1: Representasi Biner Teks

Pendefinisian data gambar dan representasi binernya dapat dilakukan dengan:

```
###################################
# dict1 = representasi biner data #
# dict2 = Kebalikan dict1         #
###################################

dict1 = {}
for i in xrange(256):
    dict1[str(i)] = '{0:08b}'.format(i)

key_dict1 = dict1.keys()
val_dict1 = dict1.values()

dict2 = {}
for j in xrange(256):
    dict2[str(val_dict1[j])] = key_dict1[j]
```

Source Code 4.4.2: Representasi Biner Gambar

Pendefinisian matriks generator G dari matriks cek H dan kode linear C:

```
########################################
# Pendefinisian Matriks Cek, Generator #
# dan Kode Linear                      #
########################################

H = matrix(GF(2),[[1,1,1,0,0,0,1,0,0,0,0,0],
                  [0,1,1,1,0,0,0,1,0,0,0,0],
                  [0,0,1,1,1,0,0,0,1,0,0,0],
                  [0,0,0,1,1,1,0,0,0,1,0,0],
                  [1,0,1,0,1,0,0,0,0,0,1,0],
                  [0,1,0,1,0,1,0,0,0,0,0,1]])
G = matrix(GF(2),[[1,0,0,0,0,0,1,0,0,0,1,0],
                  [0,1,0,0,0,0,1,1,0,0,0,1],
                  [0,0,1,0,0,0,1,1,1,0,1,0],
```

```
15                      [0,0,0,1,0,0,0,1,1,1,0,1],
16                      [0,0,0,0,1,0,0,0,1,1,1,0],
17                      [0,0,0,0,0,1,0,0,0,1,0,1]])
18  C = codes.LinearCode(generator=G)
```

Source Code 4.4.3: Konstruksi Kode Linear

Terakhir adalah pendefinisian kode Hamming (15,11) di Sage:

```
1   C = codes.HammingCode(GF(2),4)
```

Source Code 4.4.4: Konstruksi Kode Hamming

dengan 4 menyatakan orde dari kode Hamming C, yaitu $m = 4$. Kode sumber lengkap untuk setiap program simulasi dapat ditemukan pada Lampiran A atau pada situs GitHub[3].

4.5 Hasil Simulasi

Pada subbab ini akan dilakukan simulasi program. Dari hasil simulasi yang dijalankan, dapat ditentukan batas error maksimum yang dapat terjadi. Selain itu, dapat dilihat juga waktu komputasi yang dibutuhkan oleh setiap pasang encoder dan decoder.

4.5.1 Simulasi data teks

Jika digunakan kode linear (12,6), teks "Percobaan", dengan encoder matriks generator, decoder syndrome, dan n error, diperoleh hasil pengukuran waktu komputasi yang ditunjukkan pada Tabel 4.2. Perlu diingat bahwa pesan diterima mungkin berubah-ubah setiap kali menjalankan program karena jumlah error yang terjadi adalah acak. Dapat dilihat bahwa semakin besar error, data yang diterima semakin rusak. Hal ini disebabkan $d_{min}(C) = 1$.

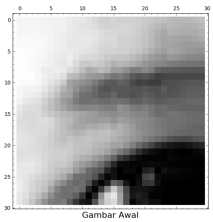
[3]https://github.com/tdtimur/simulasi-hamming-sage

Tabel 4.2: Hasil simulasi teks

n error	diterima	waktu (detik)
0	Percobaan	0.00923298
1	Percobaan	0.01316905
2	Pgscobaan	0.02014803
3	P rcobdan	0.02029514
4	UfrXobaa4	0.02278804
5	3NxcgFaan	0.02185297
6	q qE0aPol	0.02450203

4.5.2 Hasil simulasi gambar

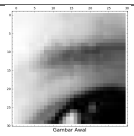
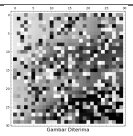
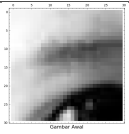
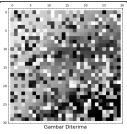
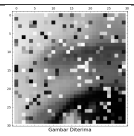
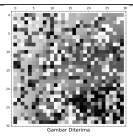
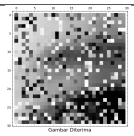
Jika digunakan kode Hamming (15,11), encoder matriks generator, decoder syndrome, n error dan gambar input



Gambar 4.2: Gambar input 30×30 piksel

diperoleh hasil pengukuran waktu komputasi yang ditunjukkan pada Tabel 4.3.

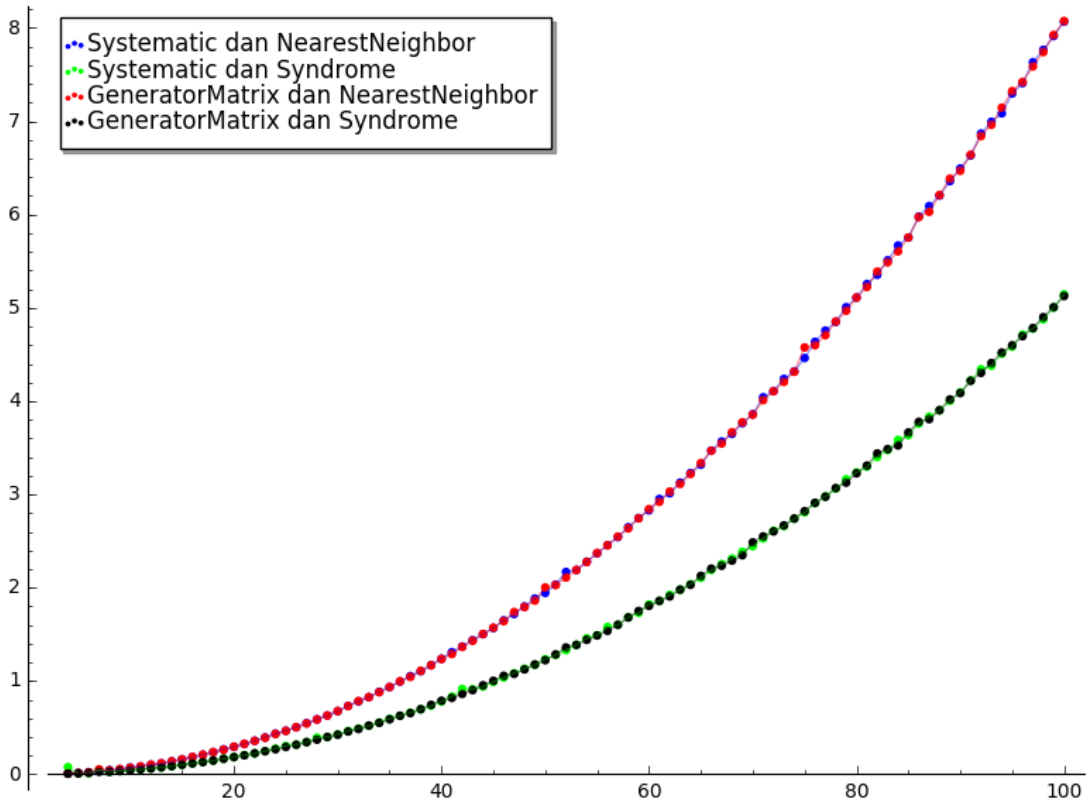
Tabel 4.3: Hasil simulasi gambar

n error	diterima	waktu	n error	diterima	waktu
0		0.558750152588	4		0.89341211319
1		0.562542200089	5		0.872886896133
2		0.771574020386	6		0.829278945923
3		0.936882972717	-		-

4.5.3 Waktu komputasi

Pada subbab ini akan dihitung waktu komputasi yang dibutuhkan pasangan encoder dan decoder untuk memproses data dengan ukuran tertentu. Grafik hasil pengukuran waktu komputasi dari kode linear (12,6) dengan data teks ditunjukkan pada Gambar 4.3, dengan sumbu x menyatakan ukuran data dalam s sepanjang $s^2 - 2s + 3$, dan sumbu y menyatakan waktu komputasi t dalam detik.

Terlihat bahwa pemilihan encoder tidak mempengaruhi waktu komputasi, tidak seperti pemilihan decoder. Decoder syndrome terlihat lebih cepat bila dibandingkan dengan decoder nearest neighbor, mulai ukuran data $s \geq 10$. Hasil ini tetap konsisten pada ukuran data hingga $s = 100$.



Gambar 4.3: Waktu komputasi kode linear dengan data teks ukuran $s \leq 100$

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan kajian dan simulasi diperoleh kesimpulan sebagai berikut.

1. Kode linear (n, k, d) C_1 dapat diubah menjadi kode linear (n, k, d') C_2 yang memiliki radius koreksi lebih besar, dengan mengubah d menjadi d' di mana $d < d'$.

2. Proses simulasi terdiri dari:

   - Input data
   - Representasi data dalam biner
   - Proses *encoding*
   - Proses transmisi
   - Proses koreksi
   - Proses *decoding*.

3. Penggunaan encoder *parity check* atau matriks generator tidak mempengaruhi kecepatan komputasi, tetapi pemilihan decoder terbukti sangat berpengaruh. Hasil pengukuran waktu komputasi menunjukkan bahwa decoder syndrome lebih cepat daripada decoder nearest neighbor.

5.2 Saran

Tugas akhir ini menggunakan kode linear dan kode Hamming untuk mensimulasikan transmisi data digital (teks

dan gambar) pada noisy channel menggunakan software SageMath. Untuk penelitian selanjutnya, diharapkan dapat mensimulasikan jenis kode lain seperti kode Reed-Solomon atau kode siklik. Selain itu, diharapkan jenis data digital yang digunakan bisa lebih bervariasi, seperti audio ataupun video.

DAFTAR PUSTAKA

Abadi, M. A. M. (2010), 'Medical Image Authentication based on Fragile Watermarking using Hamming Code', *17th Iranian Conference of Biomedical Engineering* .

Al-Bahadili, H. (2007), 'A novel lossless data compression scheme based on the error correcting Hamming codes', *Computers and Mathematics with Applications* .

Berlekamp, E. (2015), *Algebraic Coding Theory*, 2 edn, World Scientific Publishing.

Blake, I. F. dan Mullin, R. C. (1976), *An Introduction to Algebraic and Combinatorial Coding Theory*, Academic Press, Inc.

Developers, T. S. (2017), *SageMath, the Sage Mathematics Software System (Version 7.5.1)*. `http://www.sagemath.org`.

Hamming, R. W. (1986), *Coding and Information Theory*, 2 edn, Prentice Hall.

Hill, R. (1986), *A First Course in Coding Theory*, Oxford Applied Mathematics and Computing Science, Oxford University Press, Inc.

Joyner, D. dan Kim, J.-L. (2011), *Selected Unsolved Problems In Coding Theory*, Birkhäuser.

Joyner, D., Stein, W., Alexander, N. dan Johnson, N. (2017), *The Sage Reference Manual - Coding*

Theory. The Sage Reference Manual - Coding Theory. <**URL:** http://doc.sagemath.org/html/en/reference/coding/>

Kreyszig, E. (1978), *Introductory Functional Analysis With Applications*, John Wiley & Sons, Inc.

Lay, D. C., Lay, S. R. dan McDonald, J. J. (2016), *Linear Algebra and Its Applications*, 5 edn, Pearson.

Lidl, R. dan Pilz, G. (1998), *Applied Abstract Algebra*, 2 edn, Springer Verlag.

Mstafa, R. J. dan Elleithy, K. M. (2010), 'A Highly Secure Video Steganography using Hamming Code (7,4)', *IEEE Information Theory Workshop* .

Neubauer, A., Freudenberger, J. dan Kühn, V. (2007), *Coding Theory: Algorithms, Architectures, and Applications*, John Wiley & Sons, Inc.

Pinter, C. C. (1982), *A Book of Abstract Algebra*, McGraw-Hill, Inc.

Rotman, J. J. (2002), *Advanced Modern Algebra*, 1 edn, Prentice Hall.

Rurik, W. dan Mazumdar, A. (2016), 'Hamming Codes as Error-Reducing Codes', *IEEE Information Theory Workshop* .

Shannon, C. E. dan Weaver, W. (1964), *The Mathematical Theory Of Communication*, The University Of Illinois Press.

Stein, W. dan Grout, J. (2017), *The Sage Reference Manual - Interact Functions in Sage Notebook*. The Sage

Reference Manual - Interact Functions in Sage Notebook. <**URL:** http://doc.sagemath.org/html/en/reference/notebook/sagenb/notebook/interact.html>

Team, P. C. (2017*a*), *Python: A dynamic, open source programming language*, Python Software Foundation. http://www.python.org.

Team, T. S. D. (2017*b*), *The Sage Reference Manual*. <**URL:** http://doc.sagemath.org>

LAMPIRAN

LAMPIRAN A
Kode Sumber Program Simulasi

```
##############################################
# Import modul tambahan yang akan digunakan #
##############################################

import time

########################################
# Pendefinisian Matriks Cek, Generator #
# dan Kode Linear                      #
########################################

H = matrix(GF(2),[[1,1,1,0,0,0,1,0,0,0,0,0],
                  [0,1,1,1,0,0,0,1,0,0,0,0],
                  [0,0,1,1,1,0,0,0,1,0,0,0],
                  [0,0,0,1,1,1,0,0,0,1,0,0],
                  [1,0,1,0,1,0,0,0,0,0,1,0],
                  [0,1,0,1,0,1,0,0,0,0,0,1]])
G = matrix(GF(2),[[1,0,0,0,0,0,1,0,0,0,1,0],
                  [0,1,0,0,0,0,1,1,0,0,0,1],
                  [0,0,1,0,0,0,1,1,1,0,1,0],
                  [0,0,0,1,0,0,0,1,1,1,0,1],
                  [0,0,0,0,1,0,0,0,1,1,1,0],
                  [0,0,0,0,0,1,0,0,0,1,0,1]])
C = codes.LinearCode(generator=G)

###################################
# dict1 = representasi biner data #
# dict2 = Kebalikan dict1         #
###################################

strings = ' abcdefghijklmnopqrstuvwxyzABCDEFGHI
    JKLMNOPQRSTUVWXYZ1234567890.'
dict1 = {}
```

```
for i in xrange(len(strings)):
    dict1[str(strings[i])] = '{0:06b}'.format(i)

key_dict1 = dict1.keys()
val_dict1 = dict1.values()

dict2 = {}
for j in xrange(len(strings)):
    dict2[str(val_dict1[j])] = key_dict1[j]

########################################
# Pendefinisian fungsi-fungsi pembantu #
########################################

def str2bin(string):
    return dict1[string]

def bin2vec(biner):
    return vector(GF(2),[int(k) for k in biner])

def vec2bin(vector):
    return ''.join(str(k) for k in vector)

def bin2str(biner):
    return dict2[biner]

def text2bin(t):
    return ''.join(str2bin(j) for j in t)

pretty_print(html("<h2>Kode Linear Teks</h2>"))

##########################################
# Fungsi Interact dan beberapa parameter #
# simulasi yaitu teks, encoder, decoder, #
# dan banyaknya error yang dipilih       #
##########################################

@interact
def _(Teks = 'Coba.', Encoder = selector(['Systematic'
    ,'GeneratorMatrix'], label="Encoder"), Decoder =
    selector(['NearestNeighbor','Syndrome'], label="
```

```
Decoder"), Error = (0,6,1), auto_update=False):
72      E = Error
73      channel = channels.StaticErrorRateChannel(C.
ambient_space(),(int(0),int(Error)))
74      def entradec_str(string):
75
76          ####################################
77          # hasil adalah list di mana data   #
78          # simulasi disimpan dengan         #
79          # hasil[0] encoded, hasil[1]       #
80          # transmitted, hasil[2] corrected, #
81          # hasil[3] decoded, hasil[4] pesan #
82          ####################################
83
84          hasil = ['','','','','']
85
86          ####################################
87          # Proses simulasi per karakter/blok #
88          ####################################
89
90          for s in string:
91              v = bin2vec(str2bin(s))
92              ve = C.encode(v, encoder_name=Encoder)
93              ves = vec2bin(ve)
94              vt = channel.transmit(ve)
95              vts = vec2bin(vt)
96              vd = C.decode_to_code(vt, decoder_name=
Decoder)
97              vds = vec2bin(vd)
98              vm = C.decode_to_message(vt, decoder_name=
Decoder)
99              vms = vec2bin(vm)
100             m = bin2str(vms)
101             hasil[0] += ves
102             hasil[1] += vts
103             hasil[2] += vds
104             hasil[3] += vms
105             hasil[4] += m
106         return hasil
107
108     ##################
```

```
109     # Hasil simulasi #
110     ##################
111
112     start = time.time()
113
114     koding = entradec_str(Teks)
115     kod0 = koding[0]
116     kod1 = koding[1]
117     enc1 = bin2vec(kod0)
118     tra1 = bin2vec(kod1)
119     beda = enc1 + tra1
120     count = 0
121     for k in xrange(enc1.degree()):
122         if enc1[k] != tra1[k]:
123             count += 1
124     if koding[4] == Teks:
125         print "Data Terkirim Sama Dengan Data Diterima
        \n"
126     else: print "Error. Data Tidak Sama."
127
128     end = time.time()
129
130     ##############################
131     # Menampilkan hasil simulasi #
132     ##############################
133
134     print "Waktu Komputasi:", end - start ,
135     print "s"
136
137     pretty_print(html("<br><h3>Kode Linear:</h3>"))
138     print(C)
139
140     pretty_print(html("<br><h3>Matriks Cek:</h3>"))
141     print(H)
142
143     pretty_print(html("<br><h3>Matriks Generator:</h3>
        "))
144     print(G)
145
146     pretty_print(html("<br><h3>Jarak Minimum:</h3>"))
147     print(C.minimum_distance())
```

```
148
149     pretty_print(html("<h3>Teks Awal:</h3>"))
150     print(Teks)
151
152     pretty_print(html("<br><h3>Laju Informasi:</h3>"))
153     print(C.rate())
154
155     pretty_print(html("<h3>Teks Dalam Biner:</h3>"))
156     #print(''.join(str2bin(k) for k in Teks))
157     print(text2bin(Teks))
158
159     pretty_print(html("<br><h3>Hasil Encode:</h3>"))
160     print(koding[0])
161
162     pretty_print(html("<br><h3>Hasil Setelah Transmisi
        :</h3>"))
163     print(koding[1])
164
165     pretty_print(html("<br><h3>Error Yang Terjadi:</h3
        >"))
166     print(''.join(str(k) for k in beda))
167     print "\n"
168     print "Banyaknya error:", count
169
170     pretty_print(html("<br><h3>Hasil Koreksi:</h3>"))
171     print(koding[2])
172
173     pretty_print(html("<br><h3>Hasil Decode:</h3>"))
174     print(koding[3])
175
176     pretty_print(html("<br><h3>Pesan Diterima:</h3>"))
177     print(koding[4])
```

Source Code A.1: Simulasi Teks Kode Linear

```
1  ##############################################
2  # Import modul tambahan yang akan digunakan #
3  ##############################################
4
5  import cv2 as cv2
6  import time
7  import numpy as np
8  import matplotlib as mpl
9  from matplotlib import pyplot as plt
10
11 ########################################
12 # Pendefinisian Matriks Cek, Generator #
13 # dan Kode Linear                      #
14 ########################################
15
16 H = matrix(GF(2),[[1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0],
17                   [0,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0],
18                   [0,0,1,1,1,1,0,0,0,0,1,0,0,0,0,0],
19                   [0,0,0,1,1,1,1,0,0,0,0,1,0,0,0,0],
20                   [0,0,0,0,1,1,1,1,0,0,0,0,1,0,0,0],
21                   [1,0,1,0,1,0,1,0,0,0,0,0,0,1,0,0],
22                   [0,1,0,1,0,1,0,1,0,0,0,0,0,0,1,0],
23                   [1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1]])
24 G = matrix(GF(2),[[1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1],
25                   [0,1,0,0,0,0,0,0,1,1,0,0,0,0,1,1],
26                   [0,0,1,0,0,0,0,0,1,1,1,0,0,1,0,1],
27                   [0,0,0,1,0,0,0,0,1,1,1,1,0,0,1,0],
28                   [0,0,0,0,1,0,0,0,0,1,1,1,1,1,0,1],
29                   [0,0,0,0,0,1,0,0,0,0,1,1,1,0,1,1],
30                   [0,0,0,0,0,0,1,0,0,0,0,1,1,1,0,0],
31                   [0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0]])
32 C = codes.LinearCode(generator=G)
33
34 ####################################
35 # dict1 = representasi biner data #
36 # dict2 = Kebalikan dict1         #
37 ####################################
38
39 dict1 = {}
40 for i in xrange(256):
41     dict1[str(i)] = '{0:08b}'.format(i)
```

```
42
43  key_dict1 = dict1.keys()
44  val_dict1 = dict1.values()
45
46  dict2 = {}
47  for j in xrange(256):
48      dict2[str(val_dict1[j])] = key_dict1[j]
49
50  ########################################
51  # Pendefinisian fungsi-fungsi pembantu #
52  ########################################
53
54  def bin2vec(biner):
55      return vector(GF(2),[int(k) for k in biner])
56
57  def vec2bin(vector):
58      return ''.join(str(k) for k in vector)
59
60
61  ##########################################
62  # Fungsi Interact dan beberapa parameter #
63  # simulasi yaitu teks, encoder, decoder, #
64  # dan banyaknya error yang dipilih       #
65  ##########################################
66
67  @interact
68  def _(Gambar = selector([DATA+'gem2.jpg',DATA+'gem3.
        jpg',DATA+'lenna.jpg']), Encoder = selector(['
        Systematic','GeneratorMatrix'], label="Encoder"),
        Decoder = selector(['NearestNeighbor','Syndrome'])
        , Error = (0,8,1), auto_update=False):
69      start = time.time()
70      E = Error
71      channel = channels.StaticErrorRateChannel(C.
        ambient_space(),(int(0),int(Error)))
72      gambar = cv2.imread(Gambar,0)
73      baris = len(gambar[0])
74      kolom = len(gambar)
75      def entradec_img(image):
76          ###################################
77          # hasil adalah list di mana data  #
```

```
78          # simulasi disimpan dengan          #
79          # hasil[0] encoded, hasil[1]        #
80          # transmitted, hasil[2] corrected,  #
81          # hasil[3] decoded, hasil[4] pesan  #
82          ####################################
83
84          hasil = ['','','','',0,'']
85          gambar2 = []
86          hitung1 = 0
87          hitung2 = 0
88          cache = {}
89          cache2 = {}
90
91          ##############################
92          # Proses simulasi per piksel #
93          ##############################
94
95          for x in gambar:
96              gambar2.append([])
97              for y in x:
98                  if str(y) in cache:
99                      vb = cache[str(y)]
100                 else:
101                     vb = tabel[str(y)]
102                     cache[str(y)] =  tabel[str(y)]
103                 v = bin2vec(vb)
104                 ve = C.encode(v, encoder_name=Encoder)
105                 ves = vec2bin(ve)
106                 vt = channel.transmit(ve)
107                 vts = vec2bin(vt)
108                 vd = C.decode_to_code(vt, decoder_name
    =Decoder)
109                 vds = vec2bin(vd)
110                 vm = C.decode_to_message(vt,
    decoder_name=Decoder)
111                 vms = vec2bin(vm)
112                 if str(vms) in cache2:
113                     m = int(cache2[str(vms)])
114                 else:
115                     m = int(lebat[str(vms)])
116                     cache2[str(vms)] = lebat[str(vms)]
```

```
117                 hasil[0] += ves
118                 hasil[1] += vts
119                 hasil[2] += vds
120                 hasil[3] += vms
121                 gambar2[hitung1].append(m)
122                 hasil[5] += vb
123                 hitung2 += 1
124             hitung1 += 1
125         hasil[4] = np.array(gambar2,dtype=int)
126         return hasil
127 
128     ###################
129     # Hasil simulasi #
130     ###################
131 
132     koding = entradec_img(gambar)
133     kod0 = koding[0]
134     kod1 = koding[1]
135     enc1 = bin2vec(kod0)
136     tra1 = bin2vec(kod1)
137     beda = enc1 + tra1
138     count = 0
139     for k in xrange(enc1.degree()):
140         if enc1[k] != tra1[k]:
141             count += 1
142     if np.array_equal(koding[4],gambar) == True:
143         print "Data Terkirim Sama Dengan Data Diterima
    \n"
144     else: print "Error. Data Tidak Sama."
145 
146     end = time.time()
147 
148     ##############################
149     # Menampilkan hasil simulasi #
150     ##############################
151 
152     print "Waktu Komputasi:", end - start ,
153     print "s"
154 
155     pretty_print(html("<br><h3>Kode Linear:</h3>"))
156     print(C)
```

```
157
158     pretty_print(html("<br><h3>Matriks Cek:</h3>"))
159     print(H)
160
161     pretty_print(html("<br><h3>Matriks Generator:</h3>
        "))
162     print(G)
163
164     pretty_print(html("<br><h3>Jarak Minimum:</h3>"))
165     print(C.minimum_distance())
166
167     pretty_print(html("<br><h3>Laju Informasi:</h3>"))
168     print(C.rate())
169
170     if len(koding[5]) <= 6000:
171
172         pretty_print(html("<h3>Gambar Dalam Biner:</h3
        >"))
173         print(koding[5])
174
175         pretty_print(html("<br><h3>Hasil Encode:</h3>"
        ))
176         print(koding[0])
177
178         pretty_print(html("<br><h3>Hasil Setelah
        Transmisi:</h3>"))
179         print(koding[1])
180
181         pretty_print(html("<br><h3>Error Yang Terjadi
        :</h3>"))
182         print(''.join(str(k) for k in beda))
183         print "\n"
184         print "Banyaknya error:", count
185
186         pretty_print(html("<br><h3>Hasil Koreksi:</h3>
        "))
187         print(koding[2])
188
189         pretty_print(html("<br><h3>Hasil Decode:</h3>"
        ))
190         print(koding[3])
```

```
191
192         show(matrix_plot(1.0-koding[4], axes_labels =[
        'Gambar Diterima','']))
193
194     else:
195         print '' > file(DATA+'img_bin.txt','w')
196         print '' > file(DATA+'img_encoded.txt','w')
197         print '' > file(DATA+'img_transmitted.txt','w'
        )
198         print '' > file(DATA+'img_error.txt','w')
199         print '' > file(DATA+'img_decoded.txt','w')
200         print '' > file(DATA+'img_received.txt','w')
201
202         pretty_print(html("<h3>Gambar Dalam Biner:</h3
        >"))
203         img_bin = file(DATA+'img_bin.txt','w')
204         img_bin.write(str(koding[5]))
205         pretty_print(html('<a href="/home/admin/3/data
        /img_bin.txt">img_bin.txt</a>'))
206
207         pretty_print(html("<br><h3>Hasil Encode:</h3>"
        ))
208         img_encoded = file(DATA+'img_encoded.txt','w')
209         img_encoded.write(str(koding[0]))
210         pretty_print(html('<a href="/home/admin/3/data
        /img_encoded.txt">img_encoded.txt</a>'))
211
212         pretty_print(html("<br><h3>Hasil Setelah
        Transmisi:</h3>"))
213         img_transmitted = file(DATA+'img_transmitted.
        txt','w')
214         img_transmitted.write(str(koding[1]))
215         pretty_print(html('<a href="/home/admin/3/data
        /img_transmitted.txt">img_transmitted.txt</a>'))
216
217         pretty_print(html("<br><h3>Error Yang Terjadi
        :</h3>"))
218         img_error = file(DATA+'img_error.txt','w')
219         img_error.write(str(''.join(str(k) for k in
        beda)))
220         pretty_print(html('<a href="/home/admin/3/data
```

```
/img_error.txt">img_error.txt</a>'))
221         print "Banyaknya error:", count
222 
223         pretty_print(html("<br><h3>Hasil Koreksi:</h3>
    "))
224         img_decoded = file(DATA+'img_decoded.txt','w')
225         img_decoded.write(str(koding[2]))
226         pretty_print(html('<a href="/home/admin/3/data
    /img_decoded.txt">img_decoded.txt</a>'))
227 
228         pretty_print(html("<br><h3>Hasil Decode:</h3>"
    ))
229         img_received = file(DATA+'img_received.txt','w
    ')
230         img_received.write(str(koding[3]))
231         pretty_print(html('<a href="/home/admin/3/data
    /img_received.txt">img_received.txt</a>'))
232         show(matrix_plot(1.0-gambar), axes_labels =['
    Gambar Awal',''])
233         show(matrix_plot(1.0-koding[4], axes_labels =[
    'Gambar Diterima','']))
234 
```

Source Code A.2: Simulasi Gambar Kode Linear

```
1  ##############################################
2  # Import modul tambahan yang akan digunakan #
3  ##############################################
4
5  import time
6
7  ################################
8  # Pendefinisian Kode Hamming #
9  ################################
10
11 C = codes.HammingCode(GF(2),4)
12
13 ####################################
14 # dict1 = representasi biner data #
15 # dict2 = Kebalikan dict1          #
16 ####################################
17
18 strings = ' abcdefghijklmnopqrstuvwxyzABCDEFGHI
       JKLMNOPQRSTUVWXYZ1234567890.'
19 dict1 = {}
20 for i in xrange(len(strings)):
21     dict1[str(strings[i])] = '{0:011b}'.format(i)
22
23 key_dict1 = dict1.keys()
24 val_dict1 = dict1.values()
25
26 dict2 = {}
27 for j in xrange(len(strings)):
28     dict2[str(val_dict1[j])] = key_dict1[j]
29
30 ########################################
31 # Pendefinisian fungsi-fungsi pembantu #
32 ########################################
33
34 def str2bin(string):
35     return dict1[string]
36
37 def bin2vec(biner):
38     return vector(GF(2),[int(k) for k in biner])
39
40 def vec2bin(vector):
```

```
41      return ''.join(str(k) for k in vector)
42
43  def bin2str(biner):
44      return dict2[biner]
45
46  def text2bin(t):
47      return ''.join(str2bin(j) for j in t)
48
49  pretty_print(html("<h2>Kode Hamming (15,11) Teks</h2>"
        ))
50
51  ##########################################
52  # Fungsi Interact dan beberapa parameter #
53  # simulasi yaitu teks, encoder, decoder, #
54  # dan banyaknya error yang dipilih       #
55  ##########################################
56
57  @interact
58  def _(Teks = 'Saya ingin mencoba menulis pesan dengan
        menyertakan angka seperti 12345.', Encoder =
        selector(['Systematic','ParityCheck'], label="
        Encoder"), Decoder = selector(['NearestNeighbor','
        Syndrome'], label="Decoder"), Error = (0,2,1),
        auto_update=False):
59      E = Error
60      channel = channels.StaticErrorRateChannel(C.
        ambient_space(),(int(0),int(Error)))
61      def entradec_str(string):
62
63          ####################################
64          # hasil adalah list di mana data   #
65          # simulasi disimpan dengan         #
66          # hasil[0] encoded, hasil[1]       #
67          # transmitted, hasil[2] corrected, #
68          # hasil[3] decoded, hasil[4] pesan #
69          ####################################
70
71          hasil = ['','','','','']
72
73          ####################################
74          # Proses simulasi per karakter/blok #
```

```
75          ####################################
76
77          for s in string:
78              v = bin2vec(str2bin(s))
79              ve = C.encode(v, encoder_name=Encoder)
80              ves = vec2bin(ve)
81              vt = channel.transmit(ve)
82              vts = vec2bin(vt)
83              vd = C.decode_to_code(vt, decoder_name=
        Decoder)
84              vds = vec2bin(vd)
85              vm = C.decode_to_message(vt, decoder_name=
        Decoder)
86              vms = vec2bin(vm)
87              m = bin2str(vms)
88              hasil[0] += ves
89              hasil[1] += vts
90              hasil[2] += vds
91              hasil[3] += vms
92              hasil[4] += m
93          return hasil
94
95      ##################
96      # Hasil simulasi #
97      ##################
98
99      start = time.time()
100
101     koding = entradec_str(Teks)
102     kod0 = koding[0]
103     kod1 = koding[1]
104     enc1 = bin2vec(kod0)
105     tra1 = bin2vec(kod1)
106     beda = enc1 + tra1
107     count = 0
108     for k in xrange(enc1.degree()):
109         if enc1[k] != tra1[k]:
110             count += 1
111     if koding[4] == Teks:
112         print "Data Terkirim Sama Dengan Data Diterima
        \n"
```

```
113     else: print "Error. Data Tidak Sama."
114
115     end = time.time()
116
117     ##############################
118     # Menampilkan hasil simulasi #
119     ##############################
120
121     print "Waktu Komputasi:", end - start ,
122     print "s"
123
124     pretty_print(html("<br><h3>Kode Linear:</h3>"))
125     print(C)
126
127     pretty_print(html("<br><h3>Matriks Cek:</h3>"))
128     print(C.parity_check_matrix())
129
130     pretty_print(html("<br><h3>Matriks Generator:</h3>
        "))
131     print(C.generator_matrix())
132
133     pretty_print(html("<br><h3>Jarak Minimum:</h3>"))
134     print(C.minimum_distance())
135
136     pretty_print(html("<h3>Teks Awal:</h3>"))
137     print(Teks)
138
139     pretty_print(html("<br><h3>Laju Informasi:</h3>"))
140     print(C.rate())
141
142     pretty_print(html("<h3>Teks Dalam Biner:</h3>"))
143     #print(''.join(str2bin(k) for k in Teks))
144     print(text2bin(Teks))
145
146     pretty_print(html("<br><h3>Hasil Encode:</h3>"))
147     print(koding[0])
148
149     pretty_print(html("<br><h3>Hasil Setelah Transmisi
        :</h3>"))
150     print(koding[1])
151
```

```
152     pretty_print(html("<br><h3>Error Yang Terjadi:</h3
        >"))
153     print(''.join(str(k) for k in beda))
154     print "\n"
155     print "Banyaknya error:", count
156 
157     pretty_print(html("<br><h3>Hasil Koreksi:</h3>"))
158     print(koding[2])
159 
160     pretty_print(html("<br><h3>Hasil Decode:</h3>"))
161     print(koding[3])
162 
163     pretty_print(html("<br><h3>Pesan Diterima:</h3>"))
164     print(koding[4])
165 
```

Source Code A.3: Simulasi Teks Kode Hamming

```
##############################################
# Import modul tambahan yang akan digunakan #
##############################################

import cv2 as cv2
import time
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt

###############################
# Pendefinisian Kode Hamming #
###############################

C = codes.HammingCode(GF(2),4)

####################################
# dict1 = representasi biner data #
# dict2 = Kebalikan dict1         #
####################################

dict1 = {}
for i in xrange(256):
    dict1[str(i)] = '{0:011b}'.format(i)

key_dict1 = dict1.keys()
val_dict1 = dict1.values()

dict2 = {}
for j in xrange(256):
    dict2[str(val_dict1[j])] = key_dict1[j]

########################################
# Pendefinisian fungsi-fungsi pembantu #
########################################

def bin2vec(biner):
    return vector(GF(2),[int(k) for k in biner])

def vec2bin(vector):
    return ''.join(str(k) for k in vector)
```

```
42
43  pretty_print(html("<h2>Kode Hamming (15,11) Gambar</h2
        >"))
44
45  ##########################################
46  # Fungsi Interact dan beberapa parameter #
47  # simulasi yaitu teks, encoder, decoder, #
48  # dan banyaknya error yang dipilih       #
49  ##########################################
50
51  @interact
52  def _(Gambar = selector([DATA+'gem2.jpg',DATA+'gem3.
        jpg',DATA+'lenna.jpg']), Encoder = selector(['
        Systematic','ParityCheck'], label="Encoder"),
        Decoder = selector(['NearestNeighbor','Syndrome'])
        , Error = (0,2,1), auto_update=False):
53      start = time.time()
54      E = Error
55      channel = channels.StaticErrorRateChannel(C.
        ambient_space(),(int(0),int(Error)))
56      gambar = cv2.imread(Gambar,0)
57      baris = len(gambar[0])
58      kolom = len(gambar)
59      def entradec_img(image):
60          ####################################
61          # hasil adalah list di mana data   #
62          # simulasi disimpan dengan         #
63          # hasil[0] encoded, hasil[1]       #
64          # transmitted, hasil[2] corrected, #
65          # hasil[3] decoded, hasil[4] pesan #
66          ####################################
67
68          hasil = ['','','','',0,'']
69          gambar2 = []
70          hitung1 = 0
71          hitung2 = 0
72          cache = {}
73          cache2 = {}
74
75          ##############################
76          # Proses simulasi per piksel #
```

```
77          ##############################
78
79          for x in gambar:
80              gambar2.append([])
81              for y in x:
82                  if str(y) in cache:
83                      vb = cache[str(y)]
84                  else:
85                      vb = dict1[str(y)]
86                      cache[str(y)] =  dict1[str(y)]
87                  v = bin2vec(vb)
88                  ve = C.encode(v, encoder_name=Encoder)
89                  ves = vec2bin(ve)
90                  vt = channel.transmit(ve)
91                  vts = vec2bin(vt)
92                  vd = C.decode_to_code(vt, decoder_name
    =Decoder)
93                  vds = vec2bin(vd)
94                  vm = C.decode_to_message(vt,
    decoder_name=Decoder)
95                  vms = vec2bin(vm)
96                  if str(vms) in cache2:
97                      m = int(cache2[str(vms)])
98                  else:
99                      m = int(dict2[str(vms)])
100                     cache2[str(vms)] = dict2[str(vms)]
101                 hasil[0] += ves
102                 hasil[1] += vts
103                 hasil[2] += vds
104                 hasil[3] += vms
105                 gambar2[hitung1].append(m)
106                 hasil[5] += vb
107                 hitung2 += 1
108             hitung1 += 1
109         hasil[4] = np.array(gambar2,dtype=int)
110         return hasil
111
112     ##################
113     # Hasil simulasi #
114     ##################
115
```

```
116     koding = entradec_img(gambar)
117     kod0 = koding[0]
118     kod1 = koding[1]
119     enc1 = bin2vec(kod0)
120     tra1 = bin2vec(kod1)
121     beda = enc1 + tra1
122     count = 0
123     for k in xrange(enc1.degree()):
124         if enc1[k] != tra1[k]:
125             count += 1
126     if np.array_equal(koding[4],gambar) == True:
127         print "Data Terkirim Sama Dengan Data Diterima
        \n"
128     else: print "Error. Data Tidak Sama."
129 
130     end = time.time()
131 
132     ################################
133     # Menampilkan hasil simulasi #
134     ################################
135 
136     print "Waktu Komputasi:", end - start ,
137     print "s"
138 
139     pretty_print(html("<br><h3>Kode Linear:</h3>"))
140     print(C)
141 
142     pretty_print(html("<br><h3>Matriks Cek:</h3>"))
143     print(C.parity_check_matrix())
144 
145     pretty_print(html("<br><h3>Matriks Generator:</h3>
        "))
146     print(C.generator_matrix())
147 
148     pretty_print(html("<br><h3>Jarak Minimum:</h3>"))
149     print(C.minimum_distance())
150 
151     pretty_print(html("<br><h3>Laju Informasi:</h3>"))
152     print(C.rate())
153 
154     show(matrix_plot(1.0-gambar), axes_labels =['
```

```
    Gambar Awal',''])
155
156     if len(koding[5]) <= 6000:
157
158         pretty_print(html("<h3>Gambar Dalam Biner:</h3
    >"))
159         print(koding[5])
160
161         pretty_print(html("<br><h3>Hasil Encode:</h3>"
    ))
162         print(koding[0])
163
164         pretty_print(html("<br><h3>Hasil Setelah
    Transmisi:</h3>"))
165         print(koding[1])
166
167         pretty_print(html("<br><h3>Error Yang Terjadi
    :</h3>"))
168         print(''.join(str(k) for k in beda))
169         print "\n"
170         print "Banyaknya error:", count
171
172         pretty_print(html("<br><h3>Hasil Koreksi:</h3>
    "))
173         print(koding[2])
174
175         pretty_print(html("<br><h3>Hasil Decode:</h3>"
    ))
176         print(koding[3])
177
178         show(matrix_plot(1.0-koding[4], axes_labels =[
    'Gambar Diterima',''']))
179
180     else:
181         print '' > file(DATA+'img_bin.txt','w')
182         print '' > file(DATA+'img_encoded.txt','w')
183         print '' > file(DATA+'img_transmitted.txt','w'
    )
184         print '' > file(DATA+'img_error.txt','w')
185         print '' > file(DATA+'img_decoded.txt','w')
186         print '' > file(DATA+'img_received.txt','w')
```

```
187
188         pretty_print(html("<h3>Gambar Dalam Biner:</h3
        >"))
189         img_bin = file(DATA+'img_bin.txt','w')
190         img_bin.write(str(koding[5]))
191         pretty_print(html('<a href="/home/admin/7/data
        /img_bin.txt">img_bin.txt</a>'))
192
193         pretty_print(html("<br><h3>Hasil Encode:</h3>"
        ))
194         img_encoded = file(DATA+'img_encoded.txt','w')
195         img_encoded.write(str(koding[0]))
196         pretty_print(html('<a href="/home/admin/7/data
        /img_encoded.txt">img_encoded.txt</a>'))
197
198         pretty_print(html("<br><h3>Hasil Setelah
        Transmisi:</h3>"))
199         img_transmitted = file(DATA+'img_transmitted.
        txt','w')
200         img_transmitted.write(str(koding[1]))
201         pretty_print(html('<a href="/home/admin/7/data
        /img_transmitted.txt">img_transmitted.txt</a>'))
202
203         pretty_print(html("<br><h3>Error Yang Terjadi
        (Hamming Distance):</h3>"))
204         img_error = file(DATA+'img_error.txt','w')
205         img_error.write(str(''.join(str(k) for k in
        beda)))
206         pretty_print(html('<a href="/home/admin/7/data
        /img_error.txt">img_error.txt</a>'))
207         print "Banyaknya error:", count
208
209         pretty_print(html("<br><h3>Hasil Koreksi:</h3>
        "))
210         img_decoded = file(DATA+'img_decoded.txt','w')
211         img_decoded.write(str(koding[2]))
212         pretty_print(html('<a href="/home/admin/7/data
        /img_decoded.txt">img_decoded.txt</a>'))
213
214         pretty_print(html("<br><h3>Hasil Decode:</h3>"
        ))
```

```
215         img_received = file(DATA+'img_received.txt','w
    ')
216         img_received.write(str(koding[3]))
217         pretty_print(html('<a href="/home/admin/7/data
    /img_received.txt">img_received.txt</a>'))
218
219         show(matrix_plot(1.0-koding[4], axes_labels =[
    'Gambar Diterima',''']))
```

Source Code A.4: Simulasi Gambar Kode Hamming

LAMPIRAN B
Biodata Penulis

Lahir di Sumenep, 20 Juli 1993, penulis merupakan anak kedua dari dua bersaudara. Penulis menempuh pendidikan dasar dan menengah formal di SDN Pandian I Sumenep, SMPN 1 Sumenep, dan SMAN 1 Sumenep. Pada tahun 2011, penulis diterima menjadi mahasiswa S1 di Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember Surabaya. Di Departemen Matematika ini penulis mengambil rumpun mata kuliah (RMK) Analalisis dan Aljabar. Penulis pernah tergabung dalam tim Program Kreativitas Mahasiswa (PKM) tahun 2015 di bidang penelitian dengan topik kriptografi citra digital. Semasa kuliah, penulis aktif di Himpunan Mahasiswa Matematika ITS, IKAHIMATIKA Indonesia, Himpunan Mahasiswa Islam, dan pernah menjadi ketua Komunitas Open Source ITS.