

HAMMING BLOCK CODE CHANNEL ENCODER*

Ed Doering

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †


	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide ¹ module for tutorials and documentation that will help you:
	• Get started with LabVIEW
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 1

NOTE: Visit LabVIEW Setup² to learn how to adjust your own LabVIEW environment to match the settings used by the LabVIEW screencast video(s) in this module. Click the "Fullscreen" button at the lower right corner of the video player if the video does not fit properly within your browser window.

1 Summary

Channel encoding inserts additional information into a transmitted bitstream to facilitate **error detection** and **error correction** at the receiver. **Block coding** breaks up a bitstream into words of length k bits and appends check bits to form a **codeword** of length n bits. A corresponding **channel decoder** examines the complete codeword, and detects and even corrects certain types of erroneous bits caused by the channel.

In this project, develop a channel encoder using a special class of block code called a Hamming code. In a follow-on project, develop a companion channel decoder, and then evaluate the performance of the complete encoder/decoder system.

*Version 1.1: Nov 29, 2008 3:05 pm US/Central

†<http://creativecommons.org/licenses/by/2.0/>

¹"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

²"LabVIEW Setup for "Communication Systems Projects with LabVIEW"" <<http://cnx.org/content/m17319/latest/>>

2 Objectives

1. Develop an (n,k) Hamming block code channel encoder
2. Examine the behavior of the encoded bitstream before and after passing through a binary symmetric channel (BSC)
3. Learn how to use LabVIEW matrix-oriented subVIs

3 Deliverables

1. Summary write-up of your results
2. Hardcopy of all LabVIEW code that you develop (block diagrams and front panels)
3. Any plots or diagrams requested

NOTE: You can easily export LabVIEW front-panel waveform plots directly to your report. Right-click on the waveform indicator and choose "Export Simplified Image."

4 Setup

1. LabVIEW 8.5 or later version

5 Textbook Linkages

Refer to the following textbooks for additional background on the project activities of this module; see the "References" section below for publication details:

- Carlson, Crilly, and Rutledge – Ch 13 (basis for notation used in this module)
- Haykin – Ch 10
- Lathi – Ch 16
- Proakis and Salehi (FCS) – Ch 13
- Proakis and Salehi (CSE) – Ch 9
- Stern and Mahmoud – Ch 10

6 Prerequisite Modules

If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing³ which provides the foundation you need to complete this project activity, including: block diagram editing techniques, essential programming structures, subVIs, arrays, and audio.

7 Introduction

Error control coding describes a class of techniques that prepare a digital message bitstream to pass through a noisy channel so that the receiver can detect transmission errors and in some cases correct these errors.

The Figure 1 screencast video introduces error control coding, including visualization of codewords, **Hamming distance**, **minimum distance** of a code, and error detection and correction power of a code.

³*Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing*
<<http://cnx.org/content/col10440/latest/>>

Image not finished

Figure 1: [video] Error control coding basic concepts

(n,k) block codes break up a message bitstream into blocks of k bits and insert additional blocks of **checkbits**. The checkbit information permits a receiver to diagnose the received bitstream for errors, and to correct some types of errors automatically.

The Figure 2 screencast video introduces (n,k) block codes, **code rate**, the special case of **linear block codes**, and illustrates the trade-off between code rate and error control power.

Image not finished

Figure 2: [video] (n,k) block coding basic concepts

(n,k) Hamming block codes represent a popular type of block code. The Figure 3 screencast video introduces the (n,k) Hamming block code, explains how to construct the **generator matrix** to transform message blocks into codewords rate, and presents a detailed example to illustrate the encoding process.

Image not finished

Figure 3: [video] (n,k) Hamming code construction rules and example

8 Procedure

8.1 Manual calculations

Work through the Hamming code construction process by hand to lay a good foundation for developing a correct and understandable computer implementation. Write up this work on a separate page.

1. Construct two distinct $(7,4)$ Hamming code "G" matrices.
2. For each "G" matrix, calculate the codewords that emerge from the following message words: 0000, 1010, and 1111.

8.2 SubVI construction

Build the subVIs listed below. You may already have some of these available from previous projects.

Demonstrate that each of these subVIs works properly before continuing to the next part.

1. hamming_HammingCodeParameters.vi⁴
2. hamming_GeneratorMatrix.vi⁵
3. hamming_Mod2MatrixMultiply.vi⁶
4. util_BitstreamFromRandom.vi⁷
5. util_BitsToWords.vi⁸
6. util_WordsToBits.vi⁹
7. util_BinarySymmetricChannel.vi¹⁰

8.3 Hamming block code channel encoder

Review again the background theory presented earlier for the Hamming block code channel encoder, and then assemble your subVIs into a top-level application VI that creates a message bitstream, encodes the bitstream using Hamming coding, passes the bitstream through a noisy channel (the binary symmetric channel), and displays selected results of the channel encoding process.



Download the LabVIEW VI `Front_Panel_Indicators.vi`¹¹. This VI contains pre-formatted front panel indicators suitable for convenient display of binary values.

Debug your application until it works properly. Include a front-panel screenshot with hand-written annotations that demonstrates correct operation of your encoder.

9 References

1. Carlson, A. Bruce, Paul B. Crilly, and Janet C. Rutledge, "Communication Systems," 4th ed., McGraw-Hill, 2002. ISBN-13: 978-0-07-011127-1
2. Haykin, Simon. "Communication Systems," 4th ed., Wiley, 2001. ISBN-10: 0-471-17869-1
3. Lathi, Bhagwandas P., "Modern Digital and Analog Communication Systems," 3rd ed., Oxford University Press, 1998. ISBN-10: 0-19-511009-9
4. Proakis, John G., and Masoud Salehi, "Fundamentals of Communication Systems," Pearson Prentice Hall, 2005. ISBN-10: 0-13-147135-X
5. Proakis, John G., and Masoud Salehi, "Communication Systems Engineering," 2nd ed., Pearson Prentice Hall, 2002. ISBN-10: 0-13-061793-8
6. Stern, Harold P.E., and Samy A. Mahmoud, "Communication Systems," Pearson Prentice Hall, 2004. ISBN-10: 0-13-040268-0

⁴["hamming_HammingCodeParameters.vi"](http://cnx.org/content/m18441/latest/) <<http://cnx.org/content/m18441/latest/>>

⁵["hamming_GeneratorMatrix.vi"](http://cnx.org/content/m18563/latest/) <<http://cnx.org/content/m18563/latest/>>

⁶["hamming_Mod2MatrixMultiply.vi"](http://cnx.org/content/m18562/latest/) <<http://cnx.org/content/m18562/latest/>>

⁷["util_BitstreamFromRandom.vi"](http://cnx.org/content/m18528/latest/) <<http://cnx.org/content/m18528/latest/>>

⁸["util_BitsToWords.vi"](http://cnx.org/content/m18596/latest/) <<http://cnx.org/content/m18596/latest/>>

⁹["util_WordsToBits.vi"](http://cnx.org/content/m18551/latest/) <<http://cnx.org/content/m18551/latest/>>

¹⁰["util_BinarySymmetricChannel.vi"](http://cnx.org/content/m18537/latest/) <<http://cnx.org/content/m18537/latest/>>

¹¹http://cnx.org/content/m18663/latest/Front_Panel_Indicators.vi