* + 1. add the error vector thus obtained to the received vector



to correct the error (because *c* ⊕ *e* ⊕ *e =c*).

* + 1. Select the data bits from the corrected word to produce the received bit sequence



The phrase **minimal** in the third item raises the point that a double (or triple or quadruple ...) error occurring during the transmission/reception of one codeword can create the same received word as a single-bit error or no error in **another** codeword. For example, (1, 0, 0, 0, 1, 0, 1)*T*and (0, 1, 0, 0, 1, 1, 1)*T* are both codewords in the example (7,4) code. The second results when the frst one experiences three bit errors (frst, second, and sixth bits). Such an error pattern cannot be detected by our coding strategy, but such multiple error patterns are very unlikely to occur. Our receiver uses the principle of maximum probability: An error-free transmission is much more likely than one with three errors if the bit-error probability *pe* is small enough.

###### Exercise 6.28.3

How small must *pe* be so that a single-bit error is more likely to occur than a triple-bit error?

### Error-Correcting Codes: Hamming Codes

 Available under [Creative Commons-ShareAlike 4.0 International License](http://creativecommons.org/licenses/by-sa/4.0/) ([http://creativecommon](http://creativecommons.org/licenses/by-sa/4.0/) [s.org/licenses/by-sa/4.0/](http://creativecommons.org/licenses/by-sa/4.0/)).

For the (7,4) example, we have 2*N-K* − 1=7 error patterns that can be corrected. We start with single-bit error patterns, and multiply them by the parity check matrix. If we obtain unique answers, we are done; if two or more error patterns yield the same result, we can try double-bit error patterns. In our case, single-bit error patterns give a unique result.