Question 5

Checking Portion with Moser Example

Code

```
1
      % Question 5, Check
2
      % Design binary arithmetic code for ternary random variable
 3
 4
      % We will code to encode a single sequence (given in Moser) to check code
 5
       % correctness before we perform the tasks underlying the problem
 6
 7
8
9
       % Block Length, M = Sequence length to be encoded
10
11
      % U = {a,b,c} ternary RV, alpha 1 = a
                               alpha 2 = b
12
      %
                                alpha 3 = c
13
14
15
     P = \{p_a, p_b, p_c\} , p_a = p_alpha_1 = 0.5
      %
                               p b = p alpha 2 = 0.3
16
                               p_c = p_alpha_3 = 0.2
17
18
19
      % The above PMF and ordering is as given in Moser Example
20
      % a ----- alpha_1 ----- 1
21
                                             (representation)
      % b ----- alpha_2 ----- 2
22
                                            (representation)
      % c ----- alpha 3 ---- 3
23
                                            (representation)
24
25
      % M will be decided from the sequence length, which is to be encoded
     % U and PMF given
26
       % Thus encoding algorithm given in Moser can be performed
27
28
29
30
      % NOTE
31
      % We will consider the sequence to be described in terms of 1,2 and 3's
32
       % instead of a, b and c's. We could write extra code to convert the a, b
33
```

```
34
       % and c's into 1,2 and 3's but this is trivial, and seems unnecessary. Thus
35
       % a sequence such as abcca will correspond to input 12331, and we will
36
       % presume that this is the input the code will be given, to further encode
37
       % it.
38
39
40
       % sequence given in Moser = baabcabbba = 2112312221
41
42 -
       seq = [2 1 1 2 3 1 2 2 2 1];
43
       pmf = [0.5 0.3 0.2]; % pmf as given in Moser example
44 -
45
       D = 2;
46 -
47
       % Additional Note
48
49
       U = \{a,b,c\} = \{alpha 1,alpha 2,alpha 3\}
50
       % pmf = {P(alpha_1),P(alpha_2),P(alpha_3)} = {P(a),P(b),P(c)}
                                                = \{0.5, 0.3, 0.2\}
51
52
53
       % Also note a,b,c and 1,2,3 are one and the same
54
       % Arrangement of a,b,c to alpha's doesn't matter, but once fixed, should
55
56
       % stay consistent to ensure algorithm sanity
57
       % Note : r in alpha_1, alpha_2,....,alpha_r here is equal to 3
58
       % r = 3
59
60
       % alpha 1 = a = 1
61
        alpha 2 = b = 2 
62
63
       % alpha 3 = c = 3
64
65 -
       r = size(pmf, 2);
                           % alphabet size = non-zero probabilities given
66
```

```
67 -
     M = size(seq, 2);
                         % Block length = sequence length here
68
69 -
     f = zeros(1,r);
70
71 - \bigcirc \text{for i} = 2:r
72 -
     f(1,i) = sum(pmf(1,1:i-1),2);
73 -
74
75
     % pmf = {p(alpha_1),p(alpha_2),p(alpha_3)}
76
     % f = \{f(alpha 1), f(alpha 2), f(alpha 3)\}
      % f(alpha 1) = 0
77
78
     % f(alpha c) = sum pmf entries from 1 to c-1, c > 1
79
      % Since we know that f(alpha_r) = sum p(alpha_i) over i = 1 to r-1, r > 1
80
81
82 -
     p = 1;
83 -
     F = 0;
84
85 - \bigcirc \text{for } k = 1:M
86 -
         obs = seq(1,k);
87 -
         F = F + p*f(1,obs);
                                              % since observation obs = z corres
88 -
         p = p*pmf(1,obs);
     L end
89 -
90
91
      % F has F M and p has p M
92
93 -
      len = ceil((log(1/p))/(log(D))) + 1;
94
95 -
      F whole dec = F; % entirety of F, in decimal, untruncated
96
97 -
      max length = 100;
98
     F_vec_bin = zeros(1,max_length); % Two things assumed, 1. Tr
99 -
```

```
100
                                                                              2. Th
                                                         2
101
102
103
        % F vec bin is a row vector which contains the post-decimal point binary rep
104
105
        % We assume that the scope of this exercise is not general, and, the limit
106
        % of 100 on the maximum description length of the binary representation of
        % F whole dec is more than sufficient.
107
108
109
        % To code a more general case, with a possibly lengthy code, much, much
        % lengthy, one could create a different data structure in MATLAB, one that
110
        % doesn't require size to be initiated, and then append onto that
111
112
        % structure. We can then, after a certain length, end the conversion
        % program if the length exceeds beyond a certain length, length that is
113
114
        % application based.
115
116
117
        % Thus in our assumption both natural binary description length and length
        % obtained must be <= 100. This can be changed based on application by
118
        % changing the max length here, thus we can argue that the code is general
119
120
        % enough for the current exercise purpose.
121
        % Converting F whole dec into binary now
122
123
124
        % Note F whole dec will contain the actual untruncated decimal value, while
        % F which till this point is equal to F whole dec, will be used and
125
        % modified to obtain its binary representation
127
128
        % Note : Naturally, F >= 0
129
130 -
       if F >= 1
131 -
            disp('Some error in algorithm, making untruncated F in decimal greater t
132 -
            flag = 1;
```

```
133 - elseif F > 0 % F_{\text{whole_dec}} is in (0,1). We now convert this in binary.
134 -
           length flag = 0;
           flag = 2;
135 -
136 - \Box for j = 1:100
137 -
               F = 2*F;
138 -
                if F < 1
139 -
                   F vec bin(1,j) = 0;
               elseif F > 1
140 -
141 -
                   F vec bin(1,j) = 1;
142 -
                    F = F - 1;
143 -
               else
144 -
                    disp('The binary representation of untruncated decimal codeword
145 -
                    F_{\text{vec\_bin}}(1,j) = 1;
                   if j ~= 100
146 -
147 -
                        disp('The length is less than 100, of the representation')
148 -
                        F_vec_bin(1,j+1) = NaN; % to represent termination
149 -
                        length flag = 1;
150 -
151 -
                        disp('The length is exactly 100')
152 -
                       length flag = 2;
153 -
                    end
154 -
                    break
155 -
                end
156 -
          end
157 - else
158
            % Implies F = 0 (or F_whole_dec = 0)
159
            % F_vec_bin stays filled with zeros, no change
            flag = 0;
160 -
161 -
       end
162
163
164
165 % CASES
```

```
166
       % flag = 1
167
168
       % some error, F >= 1, we have a weird situation
169
170
       % flag = 0
       % F = 0. Thus F vec bin stays zero, which is correct
171
172
173
       % flag = 2
       % F is in (0,1)
174
175
       % length flag = 0
176
                    implies that full binary representation has length > 100
       %
177
           length flag = 1
                    implies this representation is of length < 100
178
179
       %
                    ending bit comes before NaN in our F vec bin
180
       %
            length flag = 2
       8
181
                    implies the representation has length exactly 100
182
183
       % NOTE :
184
       % F_whole_dec contains now the untruncated decimal representation of our
       % codeword as F may have been modified to obtain the binary representation
       % in this case
186
187
188
189 -
      if flag == 1
190 -
           disp('Some weird error')
191 -
      elseif flag == 0
192
193 -
          F bin = 0;
                                                      % F whole decimal is zero j
194
195 -
          if len > 100
196 -
               F_bin_trunc = zeros(1,len);
                                              % truncated binary code
197 -
               to add = zeros(1,len);
198 -
               to add(1, len) = 1;
```

```
199 -
                F bin final = to add + F bin trunc; % direct addition works sin
200 -
          else
201 -
               F bin trunc = F vec bin(1,1:len);
                                                    % as is, since flag is 0, F
202 -
               to add = zeros(1,len);
203 -
               to add(1, len) = 1;
204 -
               F bin final = to add + F bin trunc; % direct possible due to F
205 -
           end
206
207 -
       else
208
209 -
          if (length flag == 0 && len > 100)
210
211 -
               F bin = F vec bin;
212 -
               disp('Representation incomplete due to maximum length = 100 assumpti
213 -
                F bin trunc = zeros(1,len);
214 -
               F bin trunc(1,1:100) = F vec bin(1,1:100);
215 -
               to add = zeros(1,len);
216 -
                to add(1,len) = 1;
217 -
                F_bin_final = de2bi(bi2de(F_bin_trunc, 'left-msb')+bi2de(to_add, 'left
218
219 -
           elseif (length flag == 0 && len <= 100)
220
221 -
               F bin = F vec bin;
222 -
               F bin trunc = F vec bin(1,1:len);
223 -
               to add = zeros(1,len);
224 -
               to add(1, len) = 1;
225 -
               F bin final = de2bi(bi2de(F bin trunc, 'left-msb')+bi2de(to add, 'left
226
227
228
            % For complete representations within length 100 (including 100) of the
229
           % binary code, we presume that the system is stable and thus len in
230
           % that case will be equal to or smaller than the total representational
            % length of the binary code obtained from the untruncated decimal code
231
```

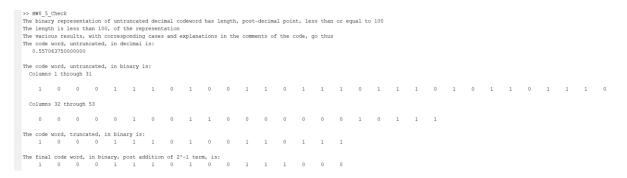
```
232
233 -
          elseif (length flag == 1)
234
235 -
               [max val red,term indx] = max(isnan(F vec bin));
236 -
               F bin = F vec bin(1,1:(term indx-1));
237 -
              F bin trunc = F bin(1,1:len);
                                             % stable system, full represe
238 -
               to add = zeros(1,len);
239 -
               to add(1, len) = 1;
240 -
               F bin final = de2bi(bi2de(F bin trunc, 'left-msb')+bi2de(to add, 'left
241
      else
242 -
243
               % length flag is 2
244
245
246 -
               F bin = F vec bin;
                                             % actual representation ends exactly
247 -
               F bin trunc = F bin(1,1:len);
248 -
               to_add = zeros(1,len);
249 -
               to add(1,len) = 1;
               F bin final = de2bi(bi2de(F_bin_trunc,'left-msb')+bi2de(to_add,'lef
250 -
251
252 -
          end
253
254
255
256 -
       end
257
258
       if flag == 1
259 -
260 -
          disp('There is nothing to be displayed')
261 -
      else
262 -
           format long
263 -
          disp('The various results, with corresponding cases and explanations in
          disp('The code word, untruncated, in decimal is:')
264 -
265 -
          disp(F whole dec)
          disp('The code word, untruncated, in binary is:')
266 -
267 -
          disp(F bin)
268 -
          disp('The code word, truncated, in binary is:')
269 -
          disp(F_bin_trunc)
270 -
           disp('The final code word, in binary, post addition of 2^-1 term, is:')
271 -
          disp(F bin final)
272 -
      end
273
```

The code has been explained in detail in the various comments.

This code takes care of a huge range of possibly unsavory situations, due to which the code length increased a little bit, despite the simplicity of the arithmetic coding algorithm.

When run, the results agreed with those given in Moser, markedly.

Results of the Check Code



Above snap is unclear, so we copy paste the result tab.

>> HW8_5_Check

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The various results, with corresponding cases and explanations in the comments of the code, go thus

The code word, untruncated, in decimal is:

0.557063750000000

The code word, untruncated, in binary is:

Columns 1 through 31

Columns 32 through 53

The code word, truncated, in binary is:

1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1

The final code word, in binary, post addition of 2^-l term, is:

1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0

As can be seen from Moser text, the final code word obtained via the code and via Moser is the same, proving that the code works correctly.

SUB-PARTS a and b for all sequence u1, u2 and u3

Code

The coding of all results done together. Answer to the question in Part b (the latter part, relating to equations in Moser) and discussion Part c done later

```
1
       % Question 5 | All parts for all codes, excluding the c) discussion and the
2
       % answer to question asked in latter part of b)
3
      % Sub-part a) and b) for all the three codewords will be done
4
5
       % simultaneously. For each code, encoding, and the various descriptions in
       % a) and b) as described, will all, be done in this single code.
6
7
8
      % Once we obtain results we will attempt to:
9
      % 1. Answer the question asked in b) (the latter part, the code results
10
      % will be obtained here itself)
11
      % 2. Discuss the results, as asked, in c)
12
      % All comments have been removed since in the check section their
13
14
      % explanatory purpose has been fulfilled, and this makes navigating code
15
      % easier
16
17
18
      % The pmf will remain same, and, constant, for all sequences
19
20
21
22
23
24 -
      seq1 = [1 3 1 2 3 2 1 2]; % u1
25
26 -
      seq2 = [1 3 1 2 3 2 1 3];
                                        8 112
27
      seq3 = [1 3 1 2 3 2 2 2];
28 -
                                        % u3
29
30 -
      seq cell = cell(1,3);
31
     seq cell{1,1} = seq1;
32 -
```

```
33 -
      seq_cell{1,2} = seq2;
 34 -
      seq_cell{1,3} = seq3;
35
36 - \bigcirc \text{for ite} = 1:3
37
38 -
          seq = seq_cell{1,ite};
 39 -
                                  % stays same for all
          pmf = [0.5 \ 0.3 \ 0.2];
 40
41 -
          D = 2;
 42
43 -
          r = size(pmf, 2);
44 -
          M = size(seq, 2);
 45 -
           f = zeros(1,r);
 46
 47 - \Box for i = 2:r
 48 -
            f(1,i) = sum(pmf(1,1:i-1),2);
 49 -
           end
50
51
 52
 53 -
          p = 1;
54 -
          F = 0;
55
56 - \bigcirc for k = 1:M
57 -
             obs = seq(1,k);
58 -
              F = F + p*f(1,obs);
59 -
              p = p*pmf(1,obs);
60 -
          end
61
           len = ceil((log(1/p))/(log(D))) + 1;
                                                                           % We
62 -
63
64 -
          F whole dec = F;
```

```
65
66 -
            max_length = 100;
67
68 -
            F vec bin = zeros(1,max length);
69
70
71
72 -
            if F >= 1
73 -
                disp('Some error in algorithm, making untruncated F in decimal great
74 -
                flag = 1;
            elseif F > 0
75 -
76 -
                length flag = 0;
                flag = 2;
77 -
                for j = 1:100
78 -
79 -
                    F = 2*F;
                    if F < 1
80 -
81 -
                         F_{\text{vec\_bin}}(1,j) = 0;
                    elseif F > 1
82 -
83 -
                        F vec bin(1,j) = 1;
84 -
                         F = F - 1;
85 -
                    else
86 -
                        disp('The binary representation of untruncated decimal code
87 -
                         F_{\text{vec\_bin}}(1,j) = 1;
88 -
                        if j ~= 100
89 -
                             disp('The length is less than 100, of the representation
90 -
                             F vec bin(1,j+1) = NaN;
91 -
                             length_flag = 1;
92 -
                        else
93 -
                             disp('The length is exactly 100')
94 -
                             length flag = 2;
95 -
                         end
96 -
                        break
```

```
97 –
                   end
 98 -
               end
99 -
            else
100 -
                flag = 0;
101 -
            end
102
103
104
           if flag == 1
105 -
106 -
                disp('Some weird error')
107 -
            elseif flag == 0
108
                F bin = 0;
109 -
110
111 -
                if len > 100
112 -
                    F bin trunc = zeros(1,len);
113 -
                    to_add = zeros(1,len);
114 -
                    to add(1, len) = 1;
115 -
                    F bin final = to add + F bin trunc;
116 -
                else
117 -
                    F_bin_trunc = F_vec_bin(1,1:len);
118 -
                    to add = zeros(1,len);
119 -
                    to add(1, len) = 1;
120 -
                    F bin final = to add + F bin trunc;
121 -
                end
122
123 -
           else
124
125 -
                if (length_flag == 0 && len > 100)
126
127 -
                    F bin = F vec bin;
                    disp('Representation incomplete due to maximum length = 100 ass
128 -
```

```
129 -
                    F_bin_trunc = zeros(1,len);
130 -
                    F bin trunc(1,1:100) = F vec bin(1,1:100);
131 -
                    to add = zeros(1,len);
132 -
                    to add(1, len) = 1;
133 -
                     F bin final = de2bi(bi2de(F bin trunc, 'left-msb')+bi2de(to add,
134
135 -
                elseif (length flag == 0 && len <= 100)
136
137 -
                     F bin = F vec bin;
138 -
                    F bin trunc = F vec bin(1,1:len);
139 -
                    to add = zeros(1,len);
140 -
                    to add(1, len) = 1;
141 -
                    F_bin_final = de2bi(bi2de(F_bin_trunc,'left-msb')+bi2de(to_add,
142
143
144
145
146 -
                elseif (length flag == 1)
147
148 -
                     [max val red,term indx] = max(isnan(F vec bin));
149 -
                     F bin = F vec bin(1,1:(term indx-1));
                    F_bin_trunc = F_bin(1,1:len);
150 -
151 -
                    to add = zeros(1,len);
152 -
                    to add(1,len) = 1;
153 -
                     F_bin_final = de2bi(bi2de(F_bin_trunc,'left-msb')+bi2de(to_add,
154
                else
155 -
156
157 -
                     F bin = F vec bin;
158 -
                    F bin trunc = F bin(1,1:len);
159 -
                    to add = zeros(1,len);
                    to add(1,len) = 1;
160 -
```

```
161 -
                     F bin final = de2bi(bi2de(F bin trunc, 'left-msb')+bi2de(to add, '
162
163 -
                 end
164
165
166
167 -
            end
168
            if flag ~= 1
169 -
170 -
                len dec = ceil((log(1/p))/(log(10))) + 1;
171 -
                 F trunc dec interim = F whole dec*(10^len dec);
                 F trunc dec = floor(F trunc dec interim);
172 -
173 -
                 F_extr_trunc_dec_interim = F_whole_dec*(10^(len_dec-1));
174 -
                 F extr trunc dec = floor(F extr trunc dec interim);
                 F final dec = (F trunc dec*(10^-len dec) + (10^-len dec))*(10^len dec
175 -
                 F final dec = cast(F final dec, 'int32');
176 -
177 -
                F trunc dec = cast(F trunc dec, 'int32');
178 -
                F extr trunc dec = cast(F extr trunc dec,'int32');
179 -
            end
180
181
182 -
            if flag == 1
183 -
                disp('There is nothing to be displayed')
184 -
            else
185 -
                format long
186 -
                disp('The results for SEQUENCE:')
187 -
                disp(ite)
188 -
                 disp('The various results, with corresponding cases and explanations
189 -
                 disp('The codeword length in binary is:')
190 -
                disp(len)
191 -
                disp('The codeword length in decimal, base 10 is:')
192 -
                disp(len dec)
193 -
                disp('The code word, untruncated, in decimal is:')
194 -
                 disp(F whole dec)
195 -
                 disp('The code word, truncated, in decimal, without adding 2^-1 term
196 -
                 disp(F trunc dec)
197 -
                 disp('The code word, truncated, in decimal, with the addition of 2^-
198 -
                 disp(F final dec)
199 -
                 disp('The code word, truncated to the 1-1 length (smaller represents
                 disp(F extr trunc dec)
200 -
201 -
                 disp('The code word, untruncated, in binary is:')
202 -
                 disp(F bin)
203 -
                 disp('The code word, truncated, in binary is:')
                disp(F_bin_trunc)
204 -
205 -
                 disp('The final code word, in binary, post addition of 2^-1 term, is
206 -
                 disp(F_bin_final)
207
208 -
                disp('NOTE: Decimal representations are converted to codewords after
209 -
            end
210
      L end
211 -
```

Result Snap

```
>> HW8_5_a_b_for_all
The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100
The length is less than 100, of the representation
The results for SEQUENCE:
The various results, with corresponding cases and explanations in the comments of the code, go thus:
The codeword length in binary is:
14
The codeword length in decimal, base 10 is:
The code word, untruncated, in decimal is: 0.438725000000000
The code word, truncated, in decimal, without adding 2^-1 term, is: 43872
The code word, truncated, in decimal, with the addition of 2^{-1} (D^-1 actually, 10^{-1} here, since D = 10) term, is: 43873
The code word, truncated to the 1-1 length (smaller representation), in decimal, obviously without any term addition, is: 4387
The code word, untruncated, in binary is:
Columns 1 through 31
     0 1 1 1
                                         0
                                               0
                                                      0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1
  Columns 32 through 52
    The code word, truncated, in binary is:
                                                0
                                                            1
                                                                   0
                                                                        1
NOTE: Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100 The length is less than 100 The length is less than 100 The results for SEQUENCE:
The various results, with corresponding cases and explanations in the comments of the code, go thus:
The codeword length in binary is:
15
The codeword length in decimal, base 10 is:
The code word, untruncated, in decimal is: 0.4388600000000000
The code word, truncated, in decimal, without adding 2^-1 term, is: 438860
 The code word, truncated, in decimal, with the addition of 2^{-1} (D^-1 actually, 10^{-1} here, since D = 10) term, is: 438861
The code word, truncated to the 1-1 length (smaller representation), in decimal, obviously without any term addition, is: 43886
The code word, untruncated, in binary is:
Columns 1 through 31
    Columns 32 through 54
    1 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0
The code word, truncated, in binary is:  0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 
NOTE: Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100
The length is less than 100, of the representation
The results for SEQUENCE:
The various results, with corresponding cases and explanations in the comments of the code, go thus: The codeword length in binary is:  
The codeword length in decimal, base 10 is:
The code word, untruncated, in decimal is: 0.439085000000000
```

Again, due to the length and width of the results, they can't be clearly seen in the Result Snaps, so we have copy pasted the entire MATLAB Results Tab to summarize the results

```
>> HW8_5_a_b_for_all
```

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The results for SEQUENCE:

1

The various results, with corresponding cases and explanations in the comments of the code, go thus:

The codeword length in binary is:

14

The codeword length in decimal, base 10 is:

5

The code word, untruncated, in decimal is:

0.438725000000000

The code word, truncated, in decimal, without adding 2^-1 term, is:

43872

The code word, truncated, in decimal, with the addition of 2^{-1} (D^-l actually, 10^{-1} here, since D = 10) term, is:

43873

The code word, truncated to the l-1 length (smaller representation), in decimal, obviously without any term addition, is:

4387

The code word, untruncated, in binary is:

Columns 1 through 31

Columns 32 through 52

The code word, truncated, in binary is:

The final code word, in binary, post addition of 2^-I term, is:

1 1 1 0 0 0 0 0 1 0 1 0 1

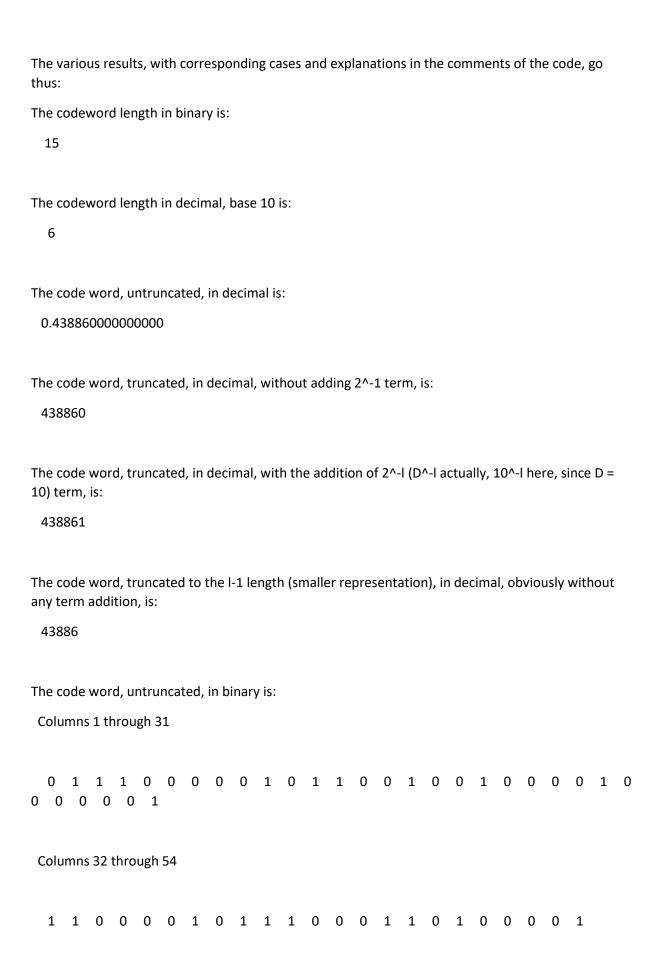
NOTE: Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The results for SEQUENCE:

2



The c	The code word, truncated, in binary is:														
0	1	1	1	0	0	0	0	0	1	0	1	1	0		0
The f	The final code word, in binary, post addition of 2^-l term, is:														
1	1	1	0	0	0	0	0	1	0	1	1	0	1		
				-											ds after removing the decimal point, only esentation
The b		-	-		ation	of u	untr	unca	ited	dec	imal	cod	lew	or	rd has length, post-decimal point, less
The length is less than 100, of the representation															
The r	The results for SEQUENCE:														
3															
The v	ario	us re	esult	ts, w	ith o	corre	espo	ndir	ng ca	ses	and	exp	lana	ati	ions in the comments of the code, go
The c	ode	wor	d ler	ngth	in b	inar	y is:								
15															
The c	ode	wor	d ler	ngth	in d	ecim	nal, l	oase	10 i	is:					
6															
The c	ode	wor	d, u	ntru	ncat	ed,	in de	ecim	al is	: :					
0.43	3908	500	000	0000)										
		wor	d, tr	runc	ated	l, in	deci	mal,	wit	hout	t add	ding	2^-	1	term, is:
439	085														
The s	، اد د						.l •			العا		ato c		.	24 1 / D4 1 to - allo - 404 1 hours of D
10) to			a, tr	unc	ateo	ı, ın	uecı	rnai,	wit	n tn	e ad	aitic	on o	T.	2^-I (D^-I actually, 10^-I here, since D =
439	086														

The c					ated	to t	he l	-1 le	ngth	n (sn	nalle	er re	pres	enta	atior	ነ), ir	ı de	cima	l, ob	viou	ısly	with	out	
439	80																							
The code word, untruncated, in binary is:																								
Colu	ımns	1 th	nrou	gh 3	1																			
0 1 1	1	1	1	0	0	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	1
Colu	imns	32 1	thro	ugh	53																			
1	0	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0	1			
The c	ode	wor	d, tr	unca	ated	, in l	oina	ry is	:															
0	1	1	1	0	0	0	0	0	1	1	0	0	1	1										
The f	inal (code	wo	rd, ii	n bir	nary	, po	st ac	lditi	on o	f 2^	-l te	rm, i	is:										
1	1	1	0	0	0	0	0	1	1	0	1	0	0											
NOTE: Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation											У													
>>																								

The results are all well summarized and described above, including all the details asked and more, and can be referred to. These results are self-explanatory.

PART b QUESTIONS

COMPARISON OF VARIOUS DECIMAL REPRESENTATIONS

IMPACT OF 5.23 AND 5.24 MOSER EQUATIONS

CAN CODEWORD FALL OUT OF THE DESIRED INTERVAL OF THE CDF?

Answer:

The reason why 2^-l is added and length made 1 more than the Shannon length is to ensure that the codes remain prefix-free, since, the PMF is not ordered.

The unordering of the PMF corresponds to a situation whereby Arithmetic Coding could give codes that are not prefix-free, if, we used the original Shannon Length and didn't add 2^-l.

This situation can be also seen from the above sequences and their results, corresponding to representations where we have 2^-I not added to the correctly truncated sequences and where we have truncated the sequences to a length equal to Shannon Length (1 less than normal Arithmetic Coding length).

Since prefix-free nature is no longer guaranteed when we truncate to a shorter length (Shannon Length) or when we don't add 2^-I to the sequence correctly truncated, the codeword may fall out of the desired cdf interval making the code not prefix-free.

Thus truncating to one more than shannon length and adding 2^-I both ensure that the arithmetic code remains prefix-free, and within correct cdf range.

PART c

RESULT DISCUSSION

Answer:

We have already discussed things in detail, as such, since the beginning, but in summary, we obtained Arithmetic Codes, both binary and decimal for different versions (truncated, untruncated, incorrectly truncated, correctly truncated without 2^-l added) and we saw how it affects the arithmetic code for different sequences for a ternary random variable.

We can say that truncating to a length more than Shannon Length and adding 2^-I is very important for the Arithmetic Codes, since doing so ensures that code stays prefix-free.