19.1

19.1 Random Codes

Question 1

The code for finding the minimum distance of a code can be found at the end of the project in the Program Listings. It works by running through all pairs of distinct codewords and computing their hamming distance. Calculating the hamming distance of two codewords of length n takes time of the order $O(n)$ and for a code of size r there are at most $\frac{r(r-1)}{2}$ distinct pairs, therefore this algorithm has total time-complexity of order $O\big(nr^2\big)$.

This code is used in conjunction with code which generates random codes of length n and size r (by simply taking r distinct elements from $\{0,1\}^n$ at random) to find the largest possible value for the minimum distance of codes with length n and size r. If we assume our code tries N different random codes, then for a given n, r, the time complexity to approximate the maximum value for the minimum distance of a code of length n and size r is $O\big(Nnr^2\big)$.

For purposes of later analysis, I will introduce a function

$$B(n,r) = \max\{d \in \mathbb{N} : \text{There exists an } [n,r,d]\text{-code}\}$$

In Appendix A.1 there is a table containing the maximum value found for the minimum distance of codes with length n and size r, for various values of n and r (i.e. approximations of $B(n,r)$ for various values of n and r).

Question 2

As the algorithm to calculate the minimum distance for a code of length n and size r is $O\big(nr^2\big)$, and we know that r must be smaller than or equal to 2^n. Thus if we assume that the size of our code is unknown, then all we can say about the algorithm to calculate minimum distance is that it has time-complexity $O(n2^n)$. This exponential in n suggests that an algorithm to find the largest possible size r of a code with length n and minimum distance d, may be exponential in n, so I should consider pairs (n,d) such that

$$A(n,d) = \max\{r \in \mathbb{N} : \text{There exists an } [n,r,d]\text{-code}\}$$

is of reasonable size. The Code to generate random codes with length n and minimum distance d can be found at the end of the project in the Program Listings. In Appendix A.2 is a table of the maximum size for various values of n and d (i.e. approximations of $A(n,d)$ for various values of n and d).

Question 3

As we can see from the graphs in Figure 1, there is a negative correlation between the information rate and error-control rate using either method of generating codes. We also see a gap that appears around the point where the error control rate lies in the interval $(0.4, 0.6)$ for the method of generating codes given n and r (i.e the case where we are given the information rate).

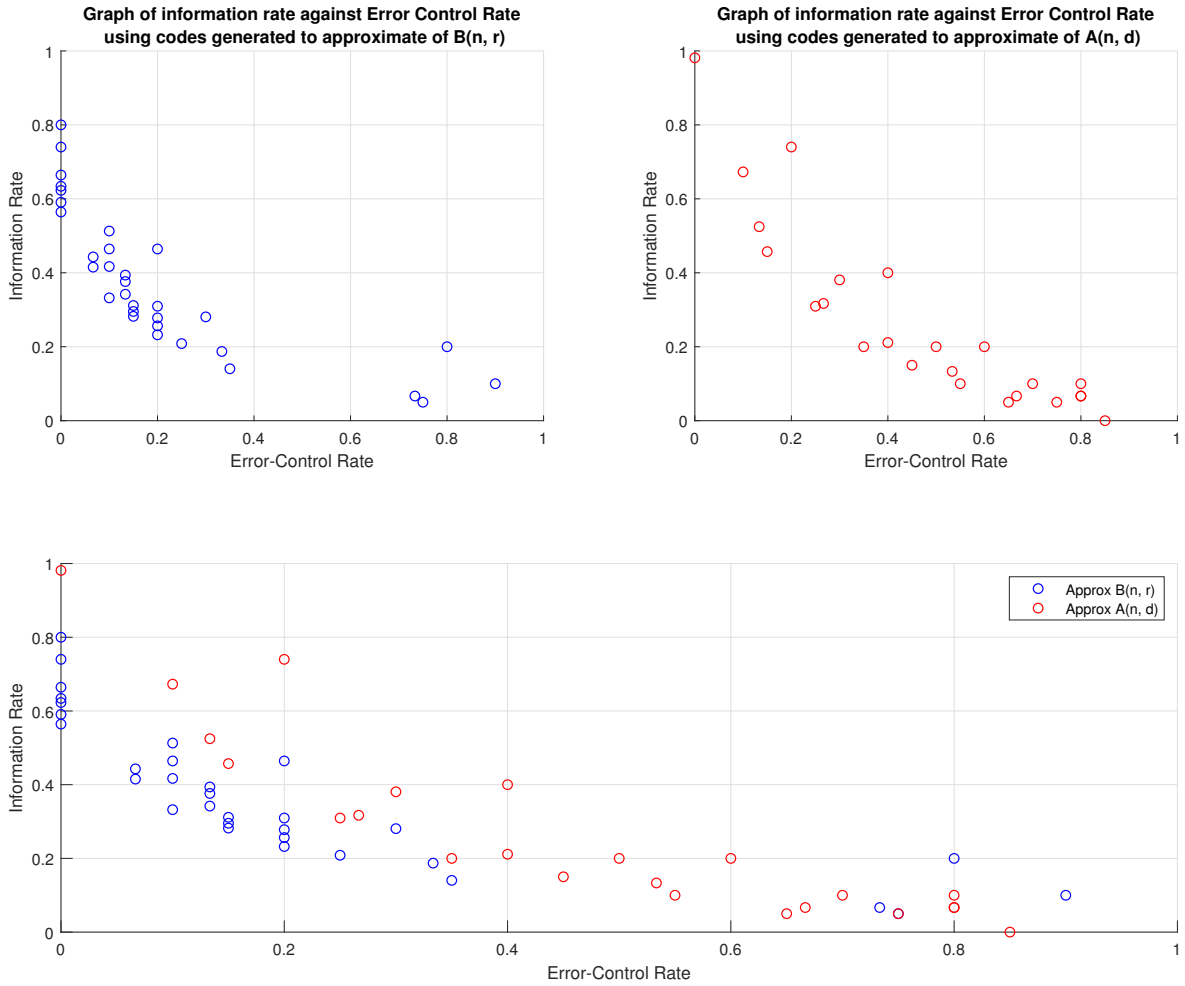

Figure 1: Plots to show the correlation between Error Control Rate and Information Rate for the random codes generated

Question 4

The code for finding the minimum distance of a linear code (of length n and rank k) works by taking as its only input the generator matrix ,$G \in \mathbb{F}_2^{k\times n}$, for the linear code and then finding the minimum weight of all non-zero codewords (i.e. finding the minimum weight of all vectors $\mathbf{c} = \mathbf{x}^T G$ where $\mathbf{x} \in \mathbb{F}_2^k \setminus \{0\}$)[1].

For the sake of future analysis define

$$\hat{A}(n,d) = \max\{k \in \mathbb{N} : \text{There exists an } (n,k,d)\text{-code}\}$$

$$\hat{B}(n,k) = \max\{d \in \mathbb{N} : \text{There exists an } (n,k,d)\text{-code}\}$$

i.e. A, B but for specifically linear codes, note that $\hat{A}(n,d) \leq \log_2 A(n,d)$ and $\hat{B}(n,k) \leq B\big(n, 2^k\big)$

The algorithm for generating linear codes given length n and rank k or length n and minimum distance d, are similar as for those in the general case, but instead of trying to generate new codewords we generate new rows for the generator matrix and test the new generator matrix.

[1] As a linear code of length n and rank k has size $2^k \leq 2^n$, this algorithm has time complexity $O\big(n2^k\big) = O(n2^n)$.

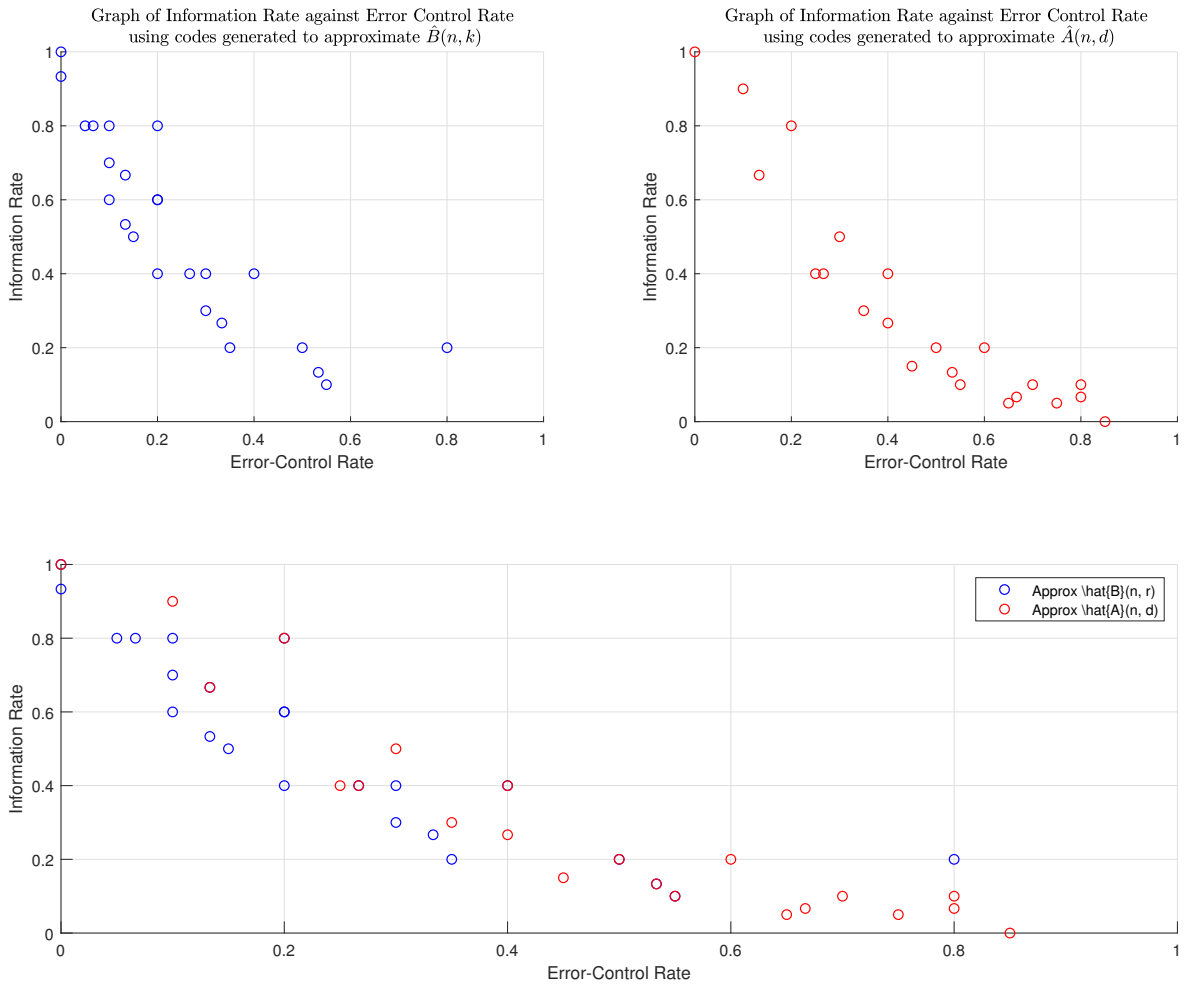

Figure 2: Plots to show the correlation between Error Control Rate and Information Rate for the random linear codes generated

In both Figure 1 and Figure 2 that there is a negative correlation between Information Rate and Error Control Rate. We can see in Figure 2 that the hole present in Figure 1 has disappeared and we can also see that there appears to be a larger amount of overlap in Figure 2 compared to Figure 1.

Question 5

Let

$$V(n,r) = \sum_{i=0}^{r} \binom{n}{i}$$

By the Gilbert–Shannon–Varshamov bound

$$A(n,d) \geq \frac{2^n}{V(n,d-1)}$$

We also know that $\delta < \frac{1}{2}$

$$\log_2 V(n, n\delta) \leq nH(\delta)$$

where $H(\delta) = -\delta \log \delta - (1-\delta)\log(1-\delta)$ is the entropy, therefore for $\frac{d-1}{n} < \frac{1}{2}$

$$\frac{\log_2 A(n,d)}{n} \geq 1 - \frac{\log_2 V(n,d-1)}{n} \geq 1 - H\left(\frac{d-1}{n}\right)$$

We also know that $A(n,d) \geq 1 \; \forall \, 0 < d \leq n \in \mathbb{N}$, so

$$\frac{\log_2 A(n,d)}{n} \geq 0$$

therefore

$$\frac{\log_2 A(n,d)}{n} \geq \begin{cases} 1 - H\left(\frac{d-1}{n}\right) & \text{if } 0 \leq \frac{d-1}{n} < \frac{1}{2} \\ 0 & \text{if } \frac{1}{2} \leq \frac{d-1}{n} < 1 \end{cases}$$

Now note by Hamming Bound

$$A(n,d) \leq \frac{2^n}{V\left(n, \left\lfloor \frac{d-1}{2} \right\rfloor\right)} \implies \frac{\log_2 A(n,d)}{n} \leq 1 - \frac{\log_2 V\left(n, \left\lfloor \frac{d-1}{2} \right\rfloor\right)}{n}$$

To bound this we note that

$$V(n,r) \geq \binom{n}{r}$$

and

$$\sqrt{2\pi}\left(\frac{n}{e}\right)^n \sqrt{n} \leq n! \leq e\left(\frac{n}{e}\right)^n \sqrt{n} \implies \frac{1}{2}\ln 2\pi + \left(n + \frac{1}{2}\right)\ln n - n \leq \ln(n!) \leq 1 + \left(n + \frac{1}{2}\right)\ln n - n$$

A bound derived Herbert Robbins "A Remark on Stirling's Formula"

So

$$\begin{aligned} \ln \binom{n}{r} &= \ln(n!) - \ln((n-r)!) - \ln(r!) \\ &\geq \left(\frac{1}{2}\ln 2\pi - 2\right) - \frac{1}{2}\ln n - \frac{1}{2}\ln\frac{r}{n} - \frac{1}{2}\ln\left(1 - \frac{r}{n}\right) - n\left[\left(1 - \frac{r}{n}\right)\ln\left(1 - \frac{r}{n}\right) + \frac{r}{n}\ln\frac{r}{n}\right] \\ &\geq -2 - \frac{1}{2}\ln n - \frac{1}{2}\ln 2 + nH\left(\frac{r}{n}\right)\ln 2 \qquad \text{Using Gibb's Inequality} \end{aligned}$$

Therefore

$$\frac{1}{n}\log_2 V(n,r) \geq \frac{1}{n}\log_2 \binom{n}{r} \geq \frac{1}{n}\left(1 - \frac{2}{\ln 2} - \log_2 n\right) + H\left(\frac{r}{n}\right)$$

Which implies

$$\frac{\log_2 A(n,d)}{n} \leq 1 - H\left(\frac{\left\lfloor \frac{d-1}{2} \right\rfloor}{n}\right) - \frac{1}{n}\left(1 - \frac{2}{\ln 2} - \log_2 n\right) \to 1 - H\left(\frac{1}{2}\delta\right)$$

Where$\delta = \lim_{n\to\infty} \frac{d-1}{n}$ i.e the given error-control rate. Thus we have bound for our information rate for large values of n, when n and d are given, i.e found approximate bounds for our graph where we used data approximating $A(n,d)$

$$\left.\begin{cases} 1 - H\left(\frac{d-1}{n}\right) & \text{if } 0 \leq \frac{d-1}{n} < \frac{1}{2} \\ 0 & \text{if } \frac{1}{2} \leq \frac{d-1}{n} < 1 \end{cases}\right\} \leq \frac{\log_2 A(n,d)}{n} \leq 1 - H\left(\frac{d-1}{2n}\right)$$

Putting these lines on our graph we see that in Figure 3 all points obey the lower bound as that result holds for all n but the upperbound is mainly satisfied by results with larger values of n (i.e. those with darker circles, the black circles represent $n = 20$)

We should note that for small n these methods are decent for finding error control codes, but as n increases the space of all possible vectors increases exponentially, so the number of checks that need to be made also grow exponentially, so we find that as n gets larger (for fixed error rate or fixed control rate), the amount of time to find an optimal code with fixed probability of success increases exponentially.

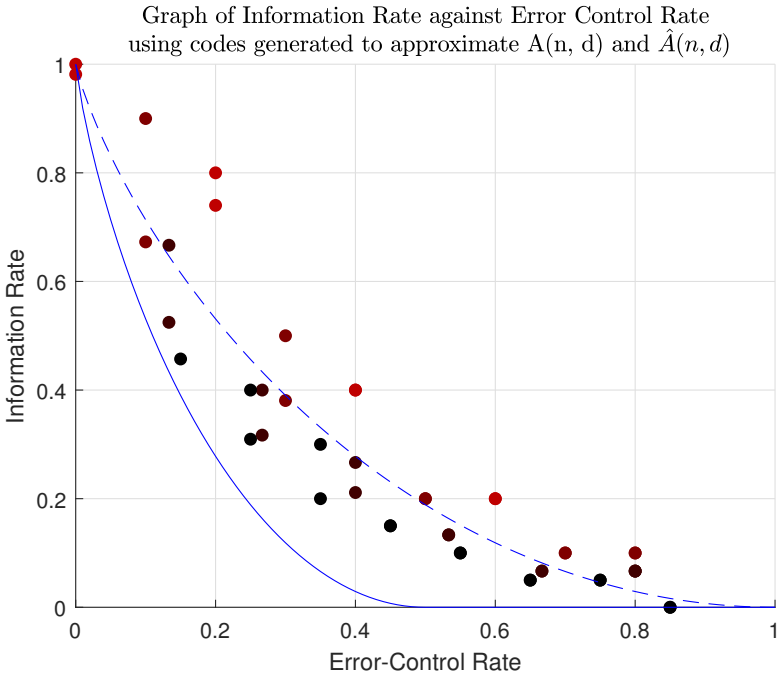

Figure 3: Plots to show the proposed bounds for codes generated using given values for n and d

Appendices

A Large Program Outputs

A.1 Table 1

n	r	Max_d
5	2	5
5	5	2
5	9	1
5	13	1
5	16	1
10	2	10
10	7	4
10	18	2
10	25	2
10	35	2
10	50	1
10	60	1
10	75	1
10	100	1
15	2	12
15	7	6
15	18	4
15	25	4
15	35	3
15	50	3
15	60	3
15	75	2
15	100	2
20	2	16
20	7	8
20	18	6
20	25	5
20	35	5
20	50	4
20	60	4
20	75	4
20	100	3

A.2 Table 2

n	d	Max_r
5	1	29
5	2	11
5	3	4
5	4	2
10	2	106
10	4	11

10	6	4
10	8	2
15	3	224
15	6	14
15	9	4
15	12	2
20	4	532
20	8	15
20	12	3
20	16	2

Programs

Code to Find the Hamming Distance Between Two Codewords

```
function d = dist(x, y)
    % Calculates Hamminng Space between 2 elements
    % of the Hamming Space
    d=0;
    for i = 1:length(x)
        d = d + abs(x(i)-y(i));
    end
end
```

Code to Find the Minimum Distance of a Code

```
function md = mindist(code)
    % Make code a cell array {} of vectors
    md = length(cell2mat(code(1)));
    for ix = 1:length(code)
        x = cell2mat(code(ix));
        for iy = ix+1:length(code)
            y = cell2mat(code(iy));
            md = min(md, dist(x,y));
        end
    end
end
```

Code to Generate a Random Code of Length n and Size r and Find its Minimum Distance

```
function d = minrandcodedist(n, r)
    code = {};
    for a = 1:r
        vec = mod(randi(2,n,1),2);
        code{end+1} = vec;
    end
    d = mindist(code);
end
```

Code to Approximate $A(n,d)$, given pairs (n,d) and number of iterations to perform per pair

```
function T = approxA(nd_pairs, trials)
    len = length(nd_pairs);
    Ns = zeros(len, 1);
    Ds = zeros(len, 1);
    Max_r = zeros(len, 1);
    for ind = 1:len
        pair = cell2mat(nd_pairs(ind));
        n = pair(1);
        d = pair(2);
        r = 0;
        for k = 1:trials
            r = max(r, minrandcodespace(n, d, 5));
```

```
        end
        Ns(ind) = n;
        Ds(ind) = d;
        Max_r(ind) = r;
        disp(join(["Finished calculations for (n, d) = (", num2str(n),...
            ", ", num2str(d), ")"], ""));
    end

    T = table(Ns, Ds, Max_r);
    T.Properties.VariableNames = {'n', 'd', 'Max_r'};
end
```

Code to Check that if a Codeword is Added to a Code, the New Code has Minimum Distance at Least d

```
function bool = checkdist(code, vec, d)
    bool = true;
    for i = 1:length(code)
        vec2 = cell2mat(code(i));
        if dist(vec, vec2) < d
            bool = false;
            break
        end
    end
end
```

Code to Generate a Random Code of Length n and Minimum Distance d and Find its Size

```
function r = minrandcodespace(n, d, tries)
    code = {zeros(n,1)};
    r = 1;
    successful = true;
    while successful
        i = 0;
        while i <= tries
            if i ~= tries
                % Tries multiple times to find a vector not in the code
                % (i.e not of distance 0) and has distance > d for all
                % elements in code
                vec = mod(randi(2,n,1),2);
                if checkdist(code, vec, d)
                    r = r + 1;
                    code{end+1} = vec;
                    break
                end
            else
                successful = false;
            end
            i = i+1;
        end
    end
end
```

Code to Approximate $B(n,r)$, given pairs (n,r) and number of iterations to perform per pair

```
function T = approxB(nr_pairs, trials)
    % pairs will be a list of pairs of (n, r)
    % Trials will be number of attempts

    len = length(nr_pairs);
    Ns = zeros(len, 1);
    Rs = zeros(len, 1);
    MinDs = zeros(len, 1);
    for ind = 1:len
        pair = cell2mat(nr_pairs(ind));
        n = pair(1);
        r = pair(2);
        d = 0;
        for k = 1:trials
            d = max(d, minrandcodedist(n,r));
        end
        Ns(ind) = n;
        Rs(ind) = r;
        MinDs(ind) = d;
        disp(join(["Finished calculations for (n, r) = (", num2str(n),...
            ", ", num2str(r), ")"], ""));
    end

    T = table(Ns, Rs, MinDs);
    T.Properties.VariableNames = {'n', 'r', 'Max_d'};
end
```

Code to Find the Minimum Distance of a Linear Code

```
function md = minlindist(gens)
    % gens will be the generator matrix
    dim = size(gens);
    % codes = zeros(2^dim(1), dim(2));
    md = Inf;
    for i = 1:(2^dim(1)-1)
        s = dec2bin(i);
        s = convertStringsToChars(s);
        vec = zeros(1, dim(1));
        for j = 1:length(s)
            vec(dim(1)+1-j) = str2double(s(length(s)+1-j));
        end
        cw = mod(vec*gens, 2);
        c = dist(cw, zeros(1, dim(2)));
        md = min(md, c);
    end
    % Note that if we get minimum distance of 0
    % then rows of the generator matrix are linearly dependent

end
```

Code to Generate a Random Linear Code of Length n and Rank k and Find its Minimum Distance

```
function d = minlinrandcodedist(n, k)
    gen = zeros(k, n);
    for a = 1:k
        vec = mod(randi(2,1,n),2);
        gen(a,:) = vec;
    end
    d = minlindist(gen);
end
```

Code to Approximate $\hat{A}(n,d)$, given pairs (n,d) and number of iterations to perform per pair

```
function T = approxLinA(nd_pairs, trials)
    len = length(nd_pairs);
    Ns = zeros(len, 1);
    Ds = zeros(len, 1);
    Max_k = zeros(len, 1);
    for ind = 1:len
        pair = cell2mat(nd_pairs(ind));
        n = pair(1);
        d = pair(2);
        k = 0;
        for p = 1:trials
            k = max(k, minlinrandcodespace(n, d, 5));
        end
        Ns(ind) = n;
        Ds(ind) = d;
        Max_k(ind) = k;
        disp(join(["Finished calculations for (n, d) = (", num2str(n),...
            ", ", num2str(d), ")"], ""));
    end

    T = table(Ns, Ds, Max_k);
    T.Properties.VariableNames = {'n', 'd', 'Max_k'};
end
```

Code to Generate a Random Linear Code of Length n and Minimum Distance d and Find its Rank

```
function k = minlinrandcodespace(n, d, tries)
    code = zeros(0,n);
    k = 0;
    successful = true;
    while successful
        i = 0;
        while i <= tries
            if i ~= tries
                % Tries multiple times to find a vector not in the code
                % (i.e not of distance 0) and has distance >= d for all
                % elements in code
                vec = mod(randi(2,1,n),2);
```

```
                while all(vec == zeros(1,n))
                    % generate a new vec if we get the zero vector
                    vec = mod(randi(2,1,n),2);
                end
                if minlindist([code;vec]) >= d
                    k = k + 1;
                    code = [code;vec];
                    break
                end
            else
                successful = false;
            end
            i = i+1;
        end
    end
end
```

Code to Approximate $\hat{B}(n,k)$, given pairs (n,k) and number of iterations to perform per pair

```
function T = approxLinB(nk_pairs, trials)
    % given n and k find d
    len = length(nk_pairs);
    Ns = zeros(len, 1);
    Ks = zeros(len, 1);
    Max_d = zeros(len, 1);
    for ind = 1:len
        pair = cell2mat(nk_pairs(ind));
        n = pair(1);
        k = pair(2);
        d = 0;
        for p = 1:trials
            d = max(d, minlinrandcodedist(n, k));
        end
        Ns(ind) = n;
        Ks(ind) = k;
        Max_d(ind) = d;
        disp(join(["Finished calculations for (n, k) = (", num2str(n),...
            ", ", num2str(k), ")"], ""));
    end

    T = table(Ns, Ks, Max_d);
    T.Properties.VariableNames = {'n', 'k', 'Max_d'};
end
```

Code to Generate Appendix A.1

```
trials = 300;

nr_pairs = ...
    {[5, 2],[5, 5],[5, 9],[5,13],[5,16],...
     [10, 2],[10,
        7],[10,18],[10,25],[10,35],[10,50],[10,60],[10,75],[10,100],...
```

```
    [15, 2],[15,
       7],[15,18],[15,25],[15,35],[15,50],[15,60],[15,75],[15,100],...
    [20, 2],[20,
       7],[20,18],[20,25],[20,35],[20,50],[20,60],[20,75],[20,100],};

T1 = approxB(nr_pairs, trials);
writetable(T1, "Q1.txt")
```

Code to Generate Appendix A.2

```
trials = 300;
nd_pairs = {[5, 1],[5, 2],[5, 3],[5, 4],...
    [10, 2],[10, 4],[10, 6],[10, 8],[10, 9],...
    [15, 3],[15, 5],[15, 7],[15,9],[15,11],[15,13],[15,13]...
    [20, 4],[20, 6],[20, 8],[20, 10],[20,12],[20,14],[20,16],[20,18]};

T2 = approxA(nd_pairs, trials);
writetable(T2, "Q2.txt")
```

Code to Generate Figure 1

```
T1 = readtable("Q1.txt");
T2 = readtable("Q2.txt");

n1 = T1{:, 'n'};
r1 = T1{:, 'r'};
d1 = T1{:, 'Max_d'};
ir1 = log2(r1)./n1;
ecr1 = (d1-1)./n1;

n2 = T2{:, 'n'};
r2 = T2{:, 'Max_r'};
d2 = T2{:, 'd'};
ir2 = log2(r2)./n2;
ecr2 = (d2-1)./n2;

figure(1);
subplot(2,2,1)
scatter(ecr1, ir1, 'blue')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
title(["Graph of information rate against Error Control Rate",...
    " using codes generated to approximate of B(n, r)"]);
grid on

subplot(2,2,2)
scatter(ecr2, ir2, 'red')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
```

```
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
title(["Graph of information rate against Error Control Rate",...
    " using codes generated to approximate of A(n, d)"]);
grid on

subplot(2,2,[3 4])
hold on
scatter(ecr1, ir1, 'blue')
scatter(ecr2, ir2, 'red')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
% Here we define B(n,r) to be the maximum of the hamming distance
% over all codes with length n and size r
xs = [0 1];

legend("Approx B(n, r)", "Approx A(n, d)")
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
grid on
hold off

set(gcf,'position',[10,10,1200,900])
print("Q3 graph.eps", '-depsc');
```

Code to Generate Figure 2

```
T3 = readtable("Q4a.txt");
T4 = readtable("Q4b.txt");

n3 = T3{:, 'n'};
k3 = T3{:, 'k'};
d3 = T3{:, 'Max_d'};
ir3 = k3./n3;
ecr3 = (d3-1)./n3;

n4 = T4{:, 'n'};
k4 = T4{:, 'Max_k'};
d4 = T4{:, 'd'};
ir4 = k4./n4;
ecr4 = (d4-1)./n4;

figure(2)
subplot(2,2,1)
scatter(ecr3, ir3, 'blue')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
title(["Graph of Information Rate against Error Control Rate",...
    " using codes generated to approximate $\hat{B}(n, k)$"], '
      Interpreter', 'latex');
grid on
```

```
subplot(2,2,2)
scatter(ecr4, ir4, 'red')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
title(["Graph of Information Rate against Error Control Rate",...
    " using codes generated to approximate $\hat{A}(n, d)$"], '
       Interpreter', 'latex');
grid on

subplot(2,2,[3 4])
hold on
scatter(ecr3, ir3, 'blue')
scatter(ecr4, ir4, 'red')
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
% Here we define B(n,r) to be the maximum of the hamming distance
% over all codes with length n and size r
xs = [0 1];


legend("Approx \hat{B}(n, r)", "Approx \hat{A}(n, d)")
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
grid on
hold off

set(gcf,'position',[10,10,1200,900])
print("Q4 graph.eps", '-depsc');
```

Code to Generate Figure 3

```
T2 = readtable("Q2.txt");
T4 = readtable("Q4b.txt");

n2 = T2{:, 'n'};
r2 = T2{:, 'Max_r'};
d2 = T2{:, 'd'};
ir2 = log2(r2)./n2;
ecr2 = (d2-1)./n2;

n4 = T4{:, 'n'};
k4 = T4{:, 'Max_k'};
d4 = T4{:, 'd'};
ir4 = k4./n4;
ecr4 = (d4-1)./n4;

ir = [ir2; ir4];
ecr = [ecr2; ecr4];
n = [n2; n4];
```

```
xs = linspace(0,1, 101);
ys1 = (1+(xs.*log2(xs)+(1-xs).*log2(1-xs))).*(xs<1/2);
ys1(1)=1;
ys1(end)=0;
ys2 = 1+(0.5*xs.*log2(0.5*xs)+(1-0.5*xs).*log2(1-0.5*xs));
ys2(1)=1;

figure(3)
scatter(ecr, ir, [], [(1-n/max(n)) zeros(length(n),2)], 'filled')
l1 = line(xs, ys1, 'Color', 'blue');
l2 = line(xs, ys2, 'Color', 'blue', 'LineStyle', '--');
axis([0 1 0 1])
ylabel('Information Rate')
xlabel('Error-Control Rate')
yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
title(["Graph of Information Rate against Error Control Rate",...
    " using codes generated to approximate A(n, d) and $\hat{A}(n, d)$"],
        'Interpreter', 'latex');
grid on
print("Q5 graph.eps", '-depsc');
```