```
& Ping_K30000_N60000_RW4_CW4_Simulate.m
% Simulate transmission and decoding of a systematic regular Ping
code.
% Copyright Brendan J Frey, January 29, 2018.
% Software written and debugged from 8.00pm to 9.04pm on January
29, 2018.

% Parameters
K=30000; % Number of data bits
N=60000; % Number of codeword bits
T=4; % Column weight of Hd
EbNo=[0.975,1.15,1.225]; % List of Gaussian noise levels (dB)
B=[2000,10000,10000]; % Number of blocks per noise level
I=20; % Number of iterations for decoding
rng(0); % Seed random number generator

% Report information, compute other parameters
if mod(N-K,T)~=0 | mod(K*T,N-K)~=0 % Check N, K and T
    error('N, K and T not compatible');
end;
fprintf('Number of information bits = %d\n',K);
fprintf('Number of transmitted bits = %d\n',N);
R=K/N;
fprintf('Rate = %f\n',R);
L=K*T; % Length of convolutional encoder
A=K*T/(N-K); % Puncturing rate of convolutional encoder
s=(2*R*10.^(EbNo/10)).^-.5; % Standard deviation of Gaussian
noise
Lf=zeros(1,L+1); % Forward messages (log-ratios)
Lb=zeros(1,L); % Backward messages (log-ratios)
Lx=zeros(1,K); % Combined messages at information bits (log-
ratios)
Lxd=zeros(1,L); % Messages sent from information bits down to
accumulator
Lxu=zeros(1,L); % Messages sent from accumulator up to
information bits

% Mapping from convolutional code to data bits, using Ping
P=zeros(1,L);
for t=1:T % Loop over sub-blocks of Hd
    P((t-1)*K+1:t*K)=randperm(K); % Random permutation for sub-
block
end;

% Simulation
ber=zeros(1,length(EbNo)); % Bit error rate
wer=zeros(1,length(EbNo)); % Word error rate
der=zeros(1,length(EbNo)); % Decoding failure rate
tic;
for j=1:length(s) % Loop over noise levels
    for b=1:B(j) % Loop over transmitted blocks
        y=randn(1,L+K)*s(j)+1; % Generate channel output assuming
```

```
+1 sent
        Lc=-2*y/s(j)^2; % Channel log-likelihood ratio
        for l=2:A Lc(l:A:L)=0; end; % Puncture convolutional code
        Lf(1)=-1e20; % Initialize forward state of Markov chain
to 0
        Lb(L)=Lc(L); % Initialize backward message to channel llr
        Lx=Lc(L+1:L+K); % Initialize information bit llr to
channel llr
        Lxu(:)=0; % Initialize messages set up to information
bits to 0
        for i=1:I % Apply I iterations of decoding
            for n=1:L % Messages sent down to accumulator
                Lxd(n)=Lx(P(n))-Lxu(n);
            end;
            for n=1:L % Forward pass
                Lf(n+1)=Lc(n)+f(Lf(n),Lxd(n));
            end;
            for n=L:-1:2 % Backward pass
                Lb(n-1)=Lc(n-1)+f(Lb(n),Lxd(n));
            end;
            Lx=Lc(L+1:L+K); % Initialize information bit llr to
channel llr
            for n=1:L % Messages sent up to information bits
                Lxu(n)=f(Lf(n),Lb(n));
                Lx(P(n))=Lx(P(n))+Lxu(n);
            end;
        end;
        xhat=1-(Lx<0);
        ber(j)=ber(j)+sum(xhat); % Update BER
        wer(j)=wer(j)+(sum(xhat)>0); % Udate WER
        der(j)=der(j)+(mean(xhat)>.1); % Update DER
    end;
    ber(j)=ber(j)/K/B(j); % Compute BER
    wer(j)=wer(j)/B(j); % Compute WER
    der(j)=der(j)/B(j); % Compute FER
    fprintf('  Eb/No=%f,  ber=%.2e,  wer=%.2e,  der=%.2e\n', ...
        EbNo(j),ber(j),wer(j),der(j));
end;
toc

% Message passing for accumulator
function c = f(a,b)
if a>b c=a+log(1+exp(b-a));
else c=b+log(1+exp(a-b));
end;
if a+b>0 c=c-(a+b+log(1+exp(-a-b)));
else c=c-log(1+exp(a+b));
end;
end
```


2

```
& Ping_Plus_MacKay_K30000_N60000_RW458_CW39_Simulate.m
% Simulate transmission and decoding of a systematic Ping code,
made
% irregular using a MacKay construction.
% Copyright Brendan J Frey, January 29, 2018.
% Software written and debugged from 9.15pm to 10.30pm on January
30, 2018.

% Parameters
K=30000; % Number of data bits
N=60000; % Number of codeword bits
EbNo=[0.7,0.775,0.85,0.95]; % List of Gaussian noise levels (dB)
B=[100,100,1000,40000]; % Number of blocks per noise level
I=40; % Number of iterations for decoding
rng(2); % Seed random number generator

% Construct irregular Ping code using MacKay-like permutation
matrices.
% Row weight is 4,5, or 8. Column weight is 3 or 9
% 1      4   1 indicates a K/6 x K/6 permutation matrix
%       44   4 is a superposition of 4 K/6 x K/6 permutation
matrices
% 1- 2- 1-
% |\ |\ |\   1- is a K/3 x K/3 permutation matrix
% 11 1- 20   |\
%  1 |\ 20
H=sparse(K,K); % Matrix that Ping calls Hd
G=K/6; % Smallest sub-block size
H=H+sparse(1:G,randperm(G),ones(1,G),K,K);
for j=1:4 H=H+sparse(1:G,5*G+randperm(G),ones(1,G),K,K); end;
for j=1:4 H=H+sparse(G+[1:G],4*G+randperm(G),ones(1,G),K,K); end;
for j=1:4 H=H+sparse(G+[1:G],5*G+randperm(G),ones(1,G),K,K); end;
H=H+sparse(2*G+[1:2*G],randperm(2*G),ones(1,2*G),K,K);
for j=1:2 H=H+sparse(2*G+[1:2*G],2*G+randperm(2*G),ones(1,2
*G),K,K); end;
H=H+sparse(2*G+[1:2*G],4*G+randperm(2*G),ones(1,2*G),K,K);
H=H+sparse(4*G+[1:G],randperm(G),ones(1,G),K,K);
H=H+sparse(4*G+[1:G],G+randperm(G),ones(1,G),K,K);
H=H+sparse(5*G+[1:G],G+randperm(G),ones(1,G),K,K);
H=H+sparse(4*G+[1:2*G],2*G+randperm(2*G),ones(1,2*G),K,K);
for j=1:2 H=H+sparse(4*G+[1:G],4*G+randperm(G),ones(1,G),K,K);
end;
for j=1:2 H=H+sparse(5*G+[1:G],4*G+randperm(G),ones(1,G),K,K);
end;
H(H(:)>1)=1;
P=mod(find(H')-1,K)+1; % Mapping from convolutional code to data
bits
L=length(P); % Length of convolutional encoder
A=zeros(1,L); A(cumsum(full(sum(H,2))))=1;% Puncturing pattern

% Report information, compute other parameters
fprintf('Number of information bits = %d\n',K);
```

```
fprintf('Number of transmitted bits = %d\n',N);
R=K/N;
fprintf('Rate = %f\n',R);
s=(2*R*10.^(EbNo/10)).^-.5; % Standard deviation of Gaussian
noise
Lf=zeros(1,L+1); % Forward messages (log-ratios)
Lb=zeros(1,L); % Backward messages (log-ratios)
Lx=zeros(1,K); % Combined messages at information bits (log-
ratios)
Lxd=zeros(1,L); % Messages sent from information bits down to
accumulator
Lxu=zeros(1,L); % Messages sent from accumulator up to
information bits

% Simulation
ber=zeros(1,length(EbNo)); % Bit error rate
wer=zeros(1,length(EbNo)); % Word error rate
der=zeros(1,length(EbNo)); % Decoding failure rate
for j=1:length(s) % Loop over noise levels
    for b=1:B(j) % Loop over transmitted blocks
        y=randn(1,L+K)*s(j)+1; % Generate channel output assuming
+1 sent
        Lc=-2*y/s(j)^2; % Channel log-likelihood ratio
        Lc(1:L)=Lc(1:L).*A; % Puncture convolutional code
        Lf(1)=-1e20; % Initialize forward state of Markov chain
to 0
        Lb(L)=Lc(L); % Initialize backward message to channel llr
        Lx=Lc(L+1:L+K); % Initialize information bit llr to
channel llr
        Lxu(:)=0; % Initialize messages set up to information
bits to 0
        for i=1:I % Apply I iterations of decoding
            for n=1:L % Messages sent down to accumulator
                Lxd(n)=Lx(P(n))-Lxu(n);
            end;
            for n=1:L % Forward pass
                Lf(n+1)=Lc(n)+f(Lf(n),Lxd(n));
            end;
            for n=L:-1:2 % Backward pass
                Lb(n-1)=Lc(n-1)+f(Lb(n),Lxd(n));
            end;
            Lx=Lc(L+1:L+K); % Initialize information bit llr to
channel llr
            for n=1:L % Messages sent up to information bits
                Lxu(n)=f(Lf(n),Lb(n));
                Lx(P(n))=Lx(P(n))+Lxu(n);
            end;
        end;
        xhat=1-(Lx<0);
        ber(j)=ber(j)+sum(xhat); % Update BER
        wer(j)=wer(j)+(sum(xhat)>0); % Udate WER
        der(j)=der(j)+(mean(xhat)>.1); % Update DER
```

```
    end;
    ber(j)=ber(j)/K/B(j); % Compute BER
    wer(j)=wer(j)/B(j); % Compute WER
    der(j)=der(j)/B(j); % Compute FER
    fprintf('  Eb/No=%f,  ber=%.2e,  wer=%.2e,  der=%.2e\n', ...
        EbNo(j),ber(j),wer(j),der(j));
end;

% Message passing for accumulator
function c = f(a,b)
if a>b c=a+log(1+exp(b-a));
else c=b+log(1+exp(a-b));
end;
if a+b>0 c=c-(a+b+log(1+exp(-a-b)));
else c=c-log(1+exp(a+b));
end;
end
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.