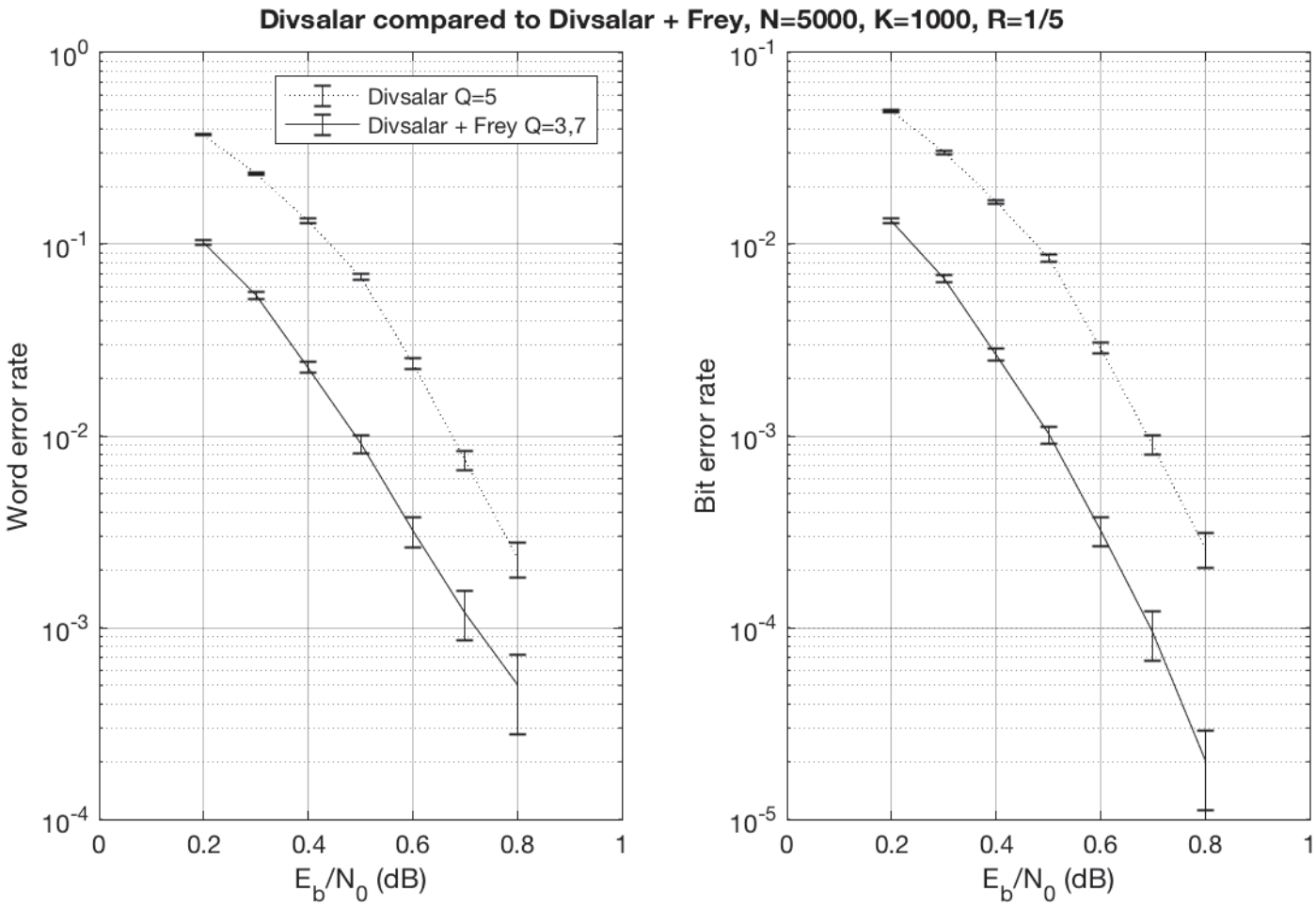
```
% Divsalar_K1000_N5000_Q5_Simulate.m
% Simulate transmission and decoding of a regular RA code.
% Copyright Brendan J Frey, January 14, 2018.
% Software written and debugged from 8.00pm to 10.45pm on January
14, 2018.

% Parameters
K=1000; % Number of information bits
EbNo=[-.8:.1:.8]; % List of Gaussian noise levels (dB)
B=10000; % Number of blocks per noise level
I=100; % Number of iterations for decoding
q=5; % Number of times to repeat information bits
Pi=repmat([1:K],[q,1]); % Mapping from repeated bits to
information bits
Pi=Pi(:);
rng(0); % Seed random number generator

% Compute other parameters, allocate memory
N=length(Pi); % Number of transmitted bits
R=K/N; % Rate
P=randperm(N); % Random permuter
s=(2*R*10.^(EbNo/10)).^-.5; % Standard deviation of Gaussian
noise
Lf=zeros(1,N+1); % Forward messages (log-ratios)
Lb=zeros(1,N); % Backward messages (log-ratios)
Lx=zeros(1,K); % Combined messages at information bits (log-
ratios)
Lxd=zeros(1,N); % Messages sent from information bits down to
accumulator
Lxu=zeros(1,N); % Messages sent from accumulator up to
information bits
ber=zeros(1,length(EbNo)); % Bit error rate
wer=zeros(1,length(EbNo)); % Word error rate
fer=zeros(1,length(EbNo)); % Failure to decode rate
fprintf('Number of information bits = %d\n',K);
fprintf('Number of transmitted bits = %d\n',N);
fprintf('Rate = %f\n',R);

% Simulation
for j=1:length(s) % Loop over noise levels
    for b=1:B % Loop over transmitted blocks
        y=randn(1,N)*s(j)+1; % Generate channel output assuming +
1 sent
        Lc=-2*y/s(j)^2; % Channel log-likelihood ratio
        Lf(1)=-1e20; % Initialize forward state of Markov chain
to 0
        Lb(N)=Lc(N); % Initialize backward message to channel llr
        Lxd(:)=0; % Initialize messages from information bits to
0
        for i=1:I % Apply I iterations of decoding
            for n=1:N % Forward pass
                Lf(n+1)=Lc(n)+f(Lf(n),Lxd(n));
```

```
            end;
            for n=N:-1:2 % Backward pass
                Lb(n-1)=Lc(n-1)+f(Lb(n),Lxd(n));
            end;
            Lx(:)=0;
            for n=1:N % Fuse messages at information bits
                Lxu(n)=f(Lf(n),Lb(n));
                Lx(Pi(P(n)))=Lx(Pi(P(n)))+Lxu(n);
            end;
            for n=1:N % Messages sent down to accumulator
                Lxd(n)=Lx(Pi(P(n)))-Lxu(n);
            end;
        end;
        xhat=1-(Lx<0); % Threhshold information bit llrs
        ber(j)=ber(j)+sum(xhat); % Update BER
        wer(j)=wer(j)+(sum(xhat)>0); % Udate WER
        fer(j)=fer(j)+(mean(xhat)>.1); % Update DER
    end;
    ber(j)=ber(j)/K/B; % Compute BER
    wer(j)=wer(j)/B; % Compute WER
    fer(j)=fer(j)/B; % Compute FER
    fprintf('  Eb/No=%f,  ber=%e,  wer=%e,  fer=%e\n', ...
        EbNo(j),ber(j),wer(j),fer(j));
end;

% Message passing for XOR function (check node, 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
```

```
% Divsalar_Plus_Frey_K1000_N5000_Q37_Simulate.m
% Simulate transmission and decoding of Divsalar RA code modified
using
% Frey's irregular construction.
% Copyright Brendan J Frey, January 21, 2018.
% Software written and debugged from 4.00pm to 4.30pm on January
21, 2018.
% Based on Divsalar_K4096_N16384_Q4_Simulate.m.

% Parameters
k=[0,0,500,0,0,0,500]; % Number of information bits with each
degree
K=sum(k); % Number of information bits
EbNo=[-.8:.1:.8]; % List of Gaussian noise levels (dB)
B=10000; % Number of blocks per noise level
I=100; % Number of iterations for decoding
rng(0); % Seed random number generator

% Compute other parameters
Pi=[]; j=0; % Mapping from repeated bits to information bits
for i=1:length(k) tmp=repmat([j+1:j+k(i)],[i,1]); Pi=[Pi;tmp(:)];
j=j+k(i); end;
N=length(Pi); % Number of transmitted bits
P=randperm(N); % Random permuter mapping from transmitted to
repeated bits
R=K/N; % Rate
s=(2*R*10.^(EbNo/10)).^-.5; % Standard deviation of Gaussian
noise
Lf=zeros(1,N+1); % Forward messages (log-ratios)
Lb=zeros(1,N); % Backward messages (log-ratios)
Lx=zeros(1,K); % Combined messages at information bits (log-
ratios)
Lxd=zeros(1,N); % Messages sent from information bits down to
accumulator
Lxu=zeros(1,N); % Messages sent from accumulator up to
information bits
ber=zeros(1,length(EbNo)); % Bit error rate
wer=zeros(1,length(EbNo)); % Word error rate
fer=zeros(1,length(EbNo)); % Failure to decode rate
fprintf('Number of information bits = %d\n',K);
fprintf('Number of transmitted bits = %d\n',N);
fprintf('Rate = %f\n',R);

% Simulation
for j=1:length(s) % Loop over noise levels
    for b=1:B % Loop over transmitted blocks
        y=randn(1,N)*s(j)+1; % Generate channel output assuming +
1 sent
        Lc=-2*y/s(j)^2; % Channel log-likelihood ratio
        Lf(1)=-1e20; % Initialize forward state of Markov chain
to 0
        Lb(N)=Lc(N); % Initialize backward message to channel llr
```

```
        Lxd(:)=0; % Initialize messages from information bits to
0
        for i=1:I % Apply I iterations of decoding
            for n=1:N % Forward pass
                Lf(n+1)=Lc(n)+f(Lf(n),Lxd(n));
            end;
            for n=N:-1:2 % Backward pass
                Lb(n-1)=Lc(n-1)+f(Lb(n),Lxd(n));
            end;
            Lx(:)=0;
            for n=1:N % % Fuse messages at information bits
                Lxu(n)=f(Lf(n),Lb(n));
                Lx(Pi(P(n)))=Lx(Pi(P(n)))+Lxu(n);
            end;
            for n=1:N % Messages sent down to accumulator
                Lxd(n)=Lx(Pi(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
        fer(j)=fer(j)+(mean(xhat)>.1); % Update DER
    end;
    ber(j)=ber(j)/K/B; % Compute BER
    wer(j)=wer(j)/B; % Compute WER
    fer(j)=fer(j)/B; % Compute FER
    fprintf('  Eb/No=%f,  ber=%e,  wer=%e,  fer=%e\n', ...
        EbNo(j),ber(j),wer(j),fer(j));
end;

% Message passing for XOR function (check node, 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.