ECE 519 - Project 2

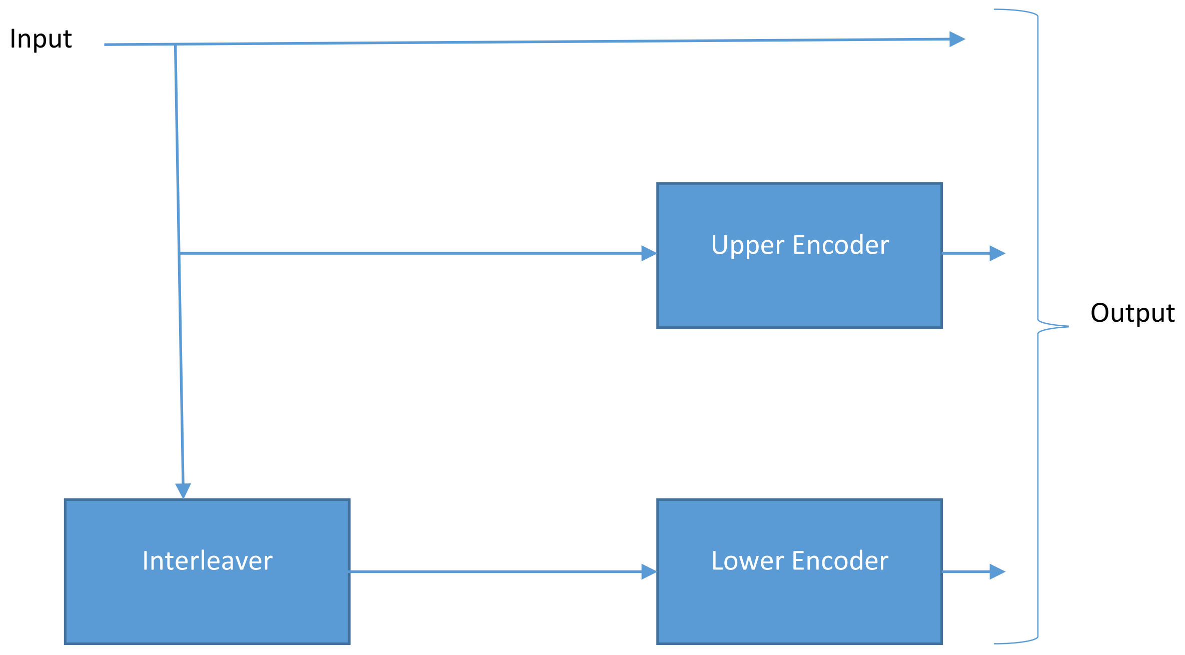
A20376688 Karthikeyan Kumar

Turbo Codes:

Turbo codes are a variant of the concatenated encoding scheme. It is constructed using two or more codes that uses different interleaved version of the same information sequence. Now the turbo codes being a concatenated code, cannot provide efficient decoding output by only making use of the hard decision. The decoding is more efficient if the decoding algorithm makes use of the soft decisions (exchanging data from the soft decisions made by each decoder unit). This procedure is iterated over many rounds in order to obtain a more reliable output.

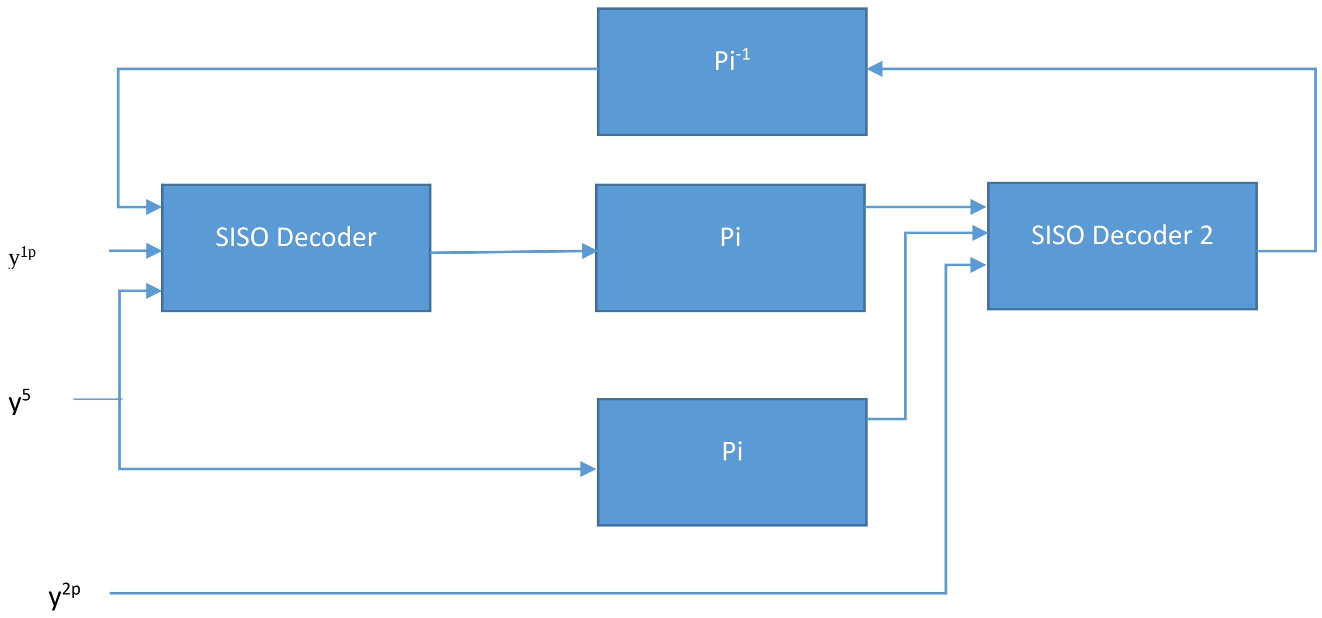
Turbo Encoder:

A turbo code is formed from the parallel concatenation of two codes separated by an interleaver. The two encoders used are normally identical. The code is in a systematic form. The main purpose of the interleaver is to randomize burst error patterns so that it can be correctly decoded. It also helps to increase the minimum distance of the turbo code. We can also use puncturing techniques in order to increase the rate of the code.



Turbo Decoder:

The decoding of turbo codes are usually done with MLD (max likelihood detector).A decoder consists of two single soft-in soft out (SISO) decoders that work iteratively. The output of one is fed into the other thereby forming a Turbo decoding iteration. The decoders process the signal, and depending on the signal amplitude they produce a soft decision output. By using the soft decision output and also the a priori probabilities we can calculate the bits to a certain degree of reliability which is then iterated again to find the final value. We either use a MAP or a LLR (Log Likelihood Ratio).



<u>Problem 1:</u>

It took Five Iterations, before the output converges.

Iteration 1:

$L_h(d)$		$L_v(d)$		$L(d)$	
-0.1	-1.5	0.1	-0.1	1.5	-1.5
-0.3	-0.2	-1.4	1.0	-1.5	1.1

Iteration 2:

$L_h(d)$		$L_v(d)$		$L(d)$	
-0.0	-1.6	1.1	-1.0	2.6	-2.5
-1.3	1.2	-1.5	1.0	-2.6	2.5

Iteration 3:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.9	-2.5	1.1	-1.0	3.5	-3.4
-1.3	1.3	-2.4	1.0	-3.5	2.6

Iteration 4:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.9	-2.5	1.1	-1.0	3.5	-3.4
-1.3	2.0	-2.4	1.0	-3.5	3.3

Iteration 5:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.9	-2.5	1.1	-1.0	3.5	-3.4
-1.3	2.0	-2.4	1.0	-3.5	3.3

Decoded output

[+1 -1 -1 +1] = [1 0 0 1]

<u>Problem 2:</u>

It took Three Iterations, before the output converges.

Iteration 1:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.61	-0.61	-0.31	0.31	3.11	-1.53
-0.23	-0.08	-2.43	1.84	-2.12	1.53

Iteration 2:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.61	-0.61	1.45	-1.3	4.87	-3.14
--1.53	1.53	-2.43	1.84	-3.88	3.14

Iteration 3:

$L_h(d)$		$L_v(d)$		$L(d)$	
0.61	-0.61	1.45	-1.3	4.87	-3.14
--1.53	1.53	-2.43	1.84	-3.88	3.14

Decoded output

[+1 -1 -1 +1] = [1 0 0 1]

```
%Computing Log Likelihood ratio as a separate function

function ret=LLR(a,b)   % Function initialisation

if a==Inf                 % compare with infinity
    ret=(-1)*b;
elseif b==Inf             % compare with infinity
    ret=(-1)*a;
elseif a==0||b==0       % setting output to zero
    ret=0;
else                        % computing output
    ret=(-1)*sign(a)*sign(b)*min(abs(a),abs(b));
end
end
```

```
clear all;
clc;
llroutput=zeros(2,2); %Initial values
temp=llroutput;
inputllr=[1.5 0.1 2.5; 0.2 0.3 2.0; 6.0 1.0 0];  %input for part a
%inputllr=[2.81 -1.23 0.61; 0.08 -0.23 1.53; 2.43 5.37 0]; %Input for part b
finalop=zeros(2,2);
i=1;
while 1
    fprintf('Iteration: %i',i);  % computation value
    temp(1,1)=LLR(inputllr(1,2)+llroutput(1,2),inputllr(1,3));
    temp(1,2)=LLR(inputllr(1,1)+llroutput(1,1),inputllr(1,3));
    temp(2,1)=LLR(inputllr(2,2)+llroutput(2,2),inputllr(2,3));
    temp(2,2)=LLR(inputllr(2,1)+llroutput(2,1),inputllr(2,3));
    llroutput=temp;
    horizontalop=llroutput; %Horizantal output
    fprintf('\nHorizontal Output: \n');
    disp(horizontalop);
    temp(1,1)=LLR(inputllr(2,1)+llroutput(2,1),inputllr(3,1));
    temp(2,1)=LLR(inputllr(1,1)+llroutput(1,1),inputllr(3,1));
    temp(1,2)=LLR(inputllr(2,2)+llroutput(2,2),inputllr(3,2));
    temp(2,2)=LLR(inputllr(1,2)+llroutput(1,2),inputllr(3,2));
    llroutput=temp;
    verticalop=llroutput;   %Vertical output
    fprintf('\nVertical Output: \n');
    disp(verticalop);
    fprintf('Final output: \n');
    finalopnew=horizontalop+verticalop+inputllr(1:2,1:2); %Final output
    disp(finalopnew);
    if(finalopnew==finalop)
        break;
    else
        finalop=finalopnew;
        i=i+1;
    end
end
```

```
Iteration: 1
Horizontal Output:
   -0.1000   -1.5000
   -0.3000   -0.2000


Vertical Output:
    0.1000   -0.1000
   -1.4000    1.0000

Final output:
    1.5000   -1.5000
   -1.5000    1.1000

Iteration: 2
Horizontal Output:
   -0.0000   -1.6000
   -1.3000    1.2000
```

```
Vertical Output:
    1.1000   -1.0000
   -1.5000    1.0000

Final output:
    2.6000   -2.5000
   -2.6000    2.5000

Iteration: 3
Horizontal Output:
    0.9000   -2.5000
   -1.3000    1.3000


Vertical Output:
    1.1000   -1.0000
   -2.4000    1.0000

Final output:
    3.5000   -3.4000
   -3.5000    2.6000

Iteration: 4
Horizontal Output:
    0.9000   -2.5000
   -1.3000    2.0000


Vertical Output:
    1.1000   -1.0000
   -2.4000    1.0000

Final output:
    3.5000   -3.4000
   -3.5000    3.3000

Iteration: 5
Horizontal Output:
    0.9000   -2.5000
   -1.3000    2.0000


Vertical Output:
    1.1000   -1.0000
   -2.4000    1.0000

Final output:
    3.5000   -3.4000
   -3.5000    3.3000
```

```
clear all;
clc;
llroutput=zeros(2,2); %Initial values
temp=llroutput;
%inputllr=[1.5 0.1 2.5; 0.2 0.3 2.0; 6.0 1.0 0];  %input for part a
inputllr=[2.81 -1.23 0.61; 0.08 -0.23 1.53; 2.43 5.37 0]; %Input for part b
finalop=zeros(2,2);
i=1;
while 1
    fprintf('Iteration: %i',i);  % computation value
    temp(1,1)=LLR(inputllr(1,2)+llroutput(1,2),inputllr(1,3));
    temp(1,2)=LLR(inputllr(1,1)+llroutput(1,1),inputllr(1,3));
    temp(2,1)=LLR(inputllr(2,2)+llroutput(2,2),inputllr(2,3));
    temp(2,2)=LLR(inputllr(2,1)+llroutput(2,1),inputllr(2,3));
    llroutput=temp;
    horizontalop=llroutput; %Horizantal output
    fprintf('\nHorizontal Output: \n');
    disp(horizontalop);
    temp(1,1)=LLR(inputllr(2,1)+llroutput(2,1),inputllr(3,1));
    temp(2,1)=LLR(inputllr(1,1)+llroutput(1,1),inputllr(3,1));
    temp(1,2)=LLR(inputllr(2,2)+llroutput(2,2),inputllr(3,2));
    temp(2,2)=LLR(inputllr(1,2)+llroutput(1,2),inputllr(3,2));
    llroutput=temp;
    verticalop=llroutput;   %Vertical output
    fprintf('\nVertical Output: \n');
    disp(verticalop);
    fprintf('Final output: \n');
    finalopnew=horizontalop+verticalop+inputllr(1:2,1:2); %Final output
    disp(finalopnew);
    if(finalopnew==finalop)
        break;
    else
        finalop=finalopnew;
        i=i+1;
    end
end
```

```
Iteration: 1
Horizontal Output:
    0.6100   -0.6100
    0.2300   -0.0800


Vertical Output:
   -0.3100    0.3100
   -2.4300    1.8400

Final output:
    3.1100   -1.5300
   -2.1200    1.5300

Iteration: 2
Horizontal Output:
    0.6100   -0.6100
   -1.5300    1.5300
```

```
Vertical Output:
    1.4500   -1.3000
   -2.4300    1.8400

Final output:
    4.8700   -3.1400
   -3.8800    3.1400

Iteration: 3
Horizontal Output:
    0.6100   -0.6100
   -1.5300    1.5300


Vertical Output:
    1.4500   -1.3000
   -2.4300    1.8400

Final output:
    4.8700   -3.1400
   -3.8800    3.1400
```

Published with MATLAB® R2014b