

LDPC and Polar codes in 5G Standard
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in
MATLAB

(Refer Slide Time: 00:16)



```
function cword = nrlldpc_encode(B,z,msg)
%B: base matrix
%z: expansion factor
%msg: message vector, length = (#cols(B)-#rows(B))*z
%cword: codeword vector, length = #cols(B)*z

[m,n] = size(B);

cword = zeros(1,n*z);
cword(1:(n-m)*z) = msg;

%double-diagonal encoding
temp = zeros(1,z);
for i = 1:4 %row 1 to 4
    for j = 1:n-m %message columns
        temp = mod(temp + mul_sh(msg((j-1)*z+1:j*z),B(i,j)),2);
    end
end
```

PROF. ANDREW THANGARAJ
IIT MADRAS

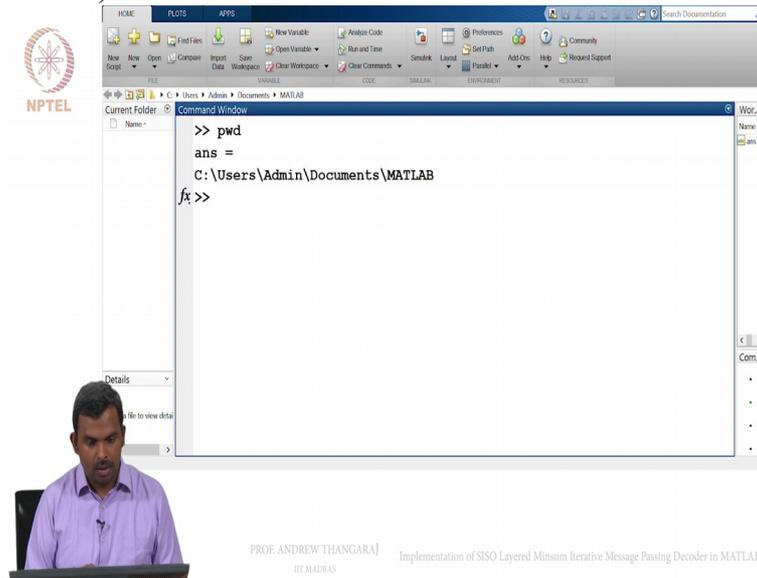
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Hello. In this lecture we will start seeing how to write a decoder for LDPC codes in the 5G standard in MATLAB, Ok. So I am going to write a quick and small MATLAB script for doing the decoding. Once again remember that we are not going for very optimized code and very efficient implementation, just to show you how to write a decoder, number 1.

Ok so in the previous MATLAB coding exercise we saw the encoder. I loaded the base matrix and hopefully you still remember all those things. You have the base matrices in a folder and then use; you know how to load it and all that. So let me quickly remind you on how that works.

So let me go to MATLAB window.

(Refer Slide Time: 01:06)



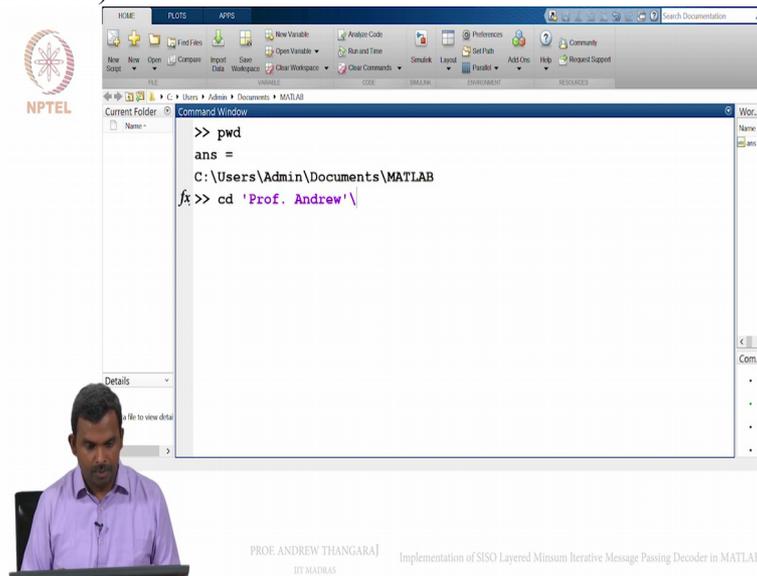
The image shows a MATLAB Command Window with the following text:

```
>> pwd
ans =
C:\Users\Admin\Documents\MATLAB
fx>>
```

Below the Command Window is a small video inset of Prof. Andrew Thangaraj. At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

I need to go

(Refer Slide Time: 01:07)



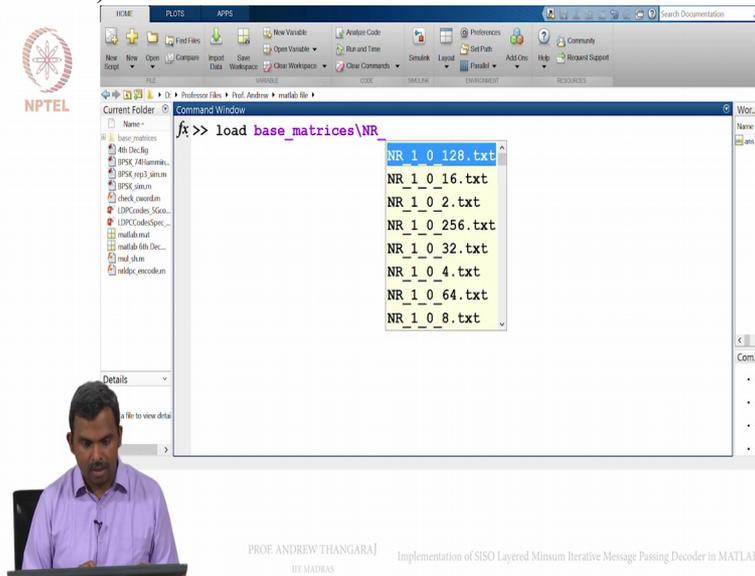
The image shows a MATLAB Command Window with the following text:

```
>> pwd
ans =
C:\Users\Admin\Documents\MATLAB
fx>> cd 'Prof. Andrew\'
```

Below the Command Window is a small video inset of Prof. Andrew Thangaraj. At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

to the correct directory, Ok. So this is my folder in which I do my coding. You can see this folder called base matrices and one can load base matrices N R,

(Refer Slide Time: 01:20)



The image shows a MATLAB Command Window with the following text:

```
f>> load base_matrices\NR
```

The output shows a list of files:

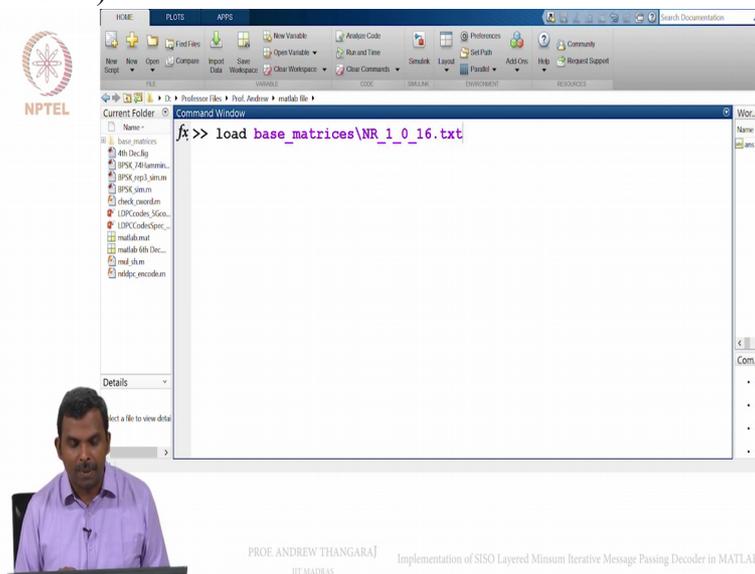
- NR_1_0_128.txt
- NR_1_0_16.txt
- NR_1_0_2.txt
- NR_1_0_256.txt
- NR_1_0_32.txt
- NR_1_0_4.txt
- NR_1_0_64.txt
- NR_1_0_8.txt

Below the screenshot, there is a small video inset of Prof. Andrew Thangaraj and a caption: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

there are whole bunch of parameters, you remember.

The first one is the base graph number, next one is that j parameter and last one is the expansion factor. So, so let us pick up 1 0 16, you could become anything so we can pick up that.

(Refer Slide Time: 01:36)



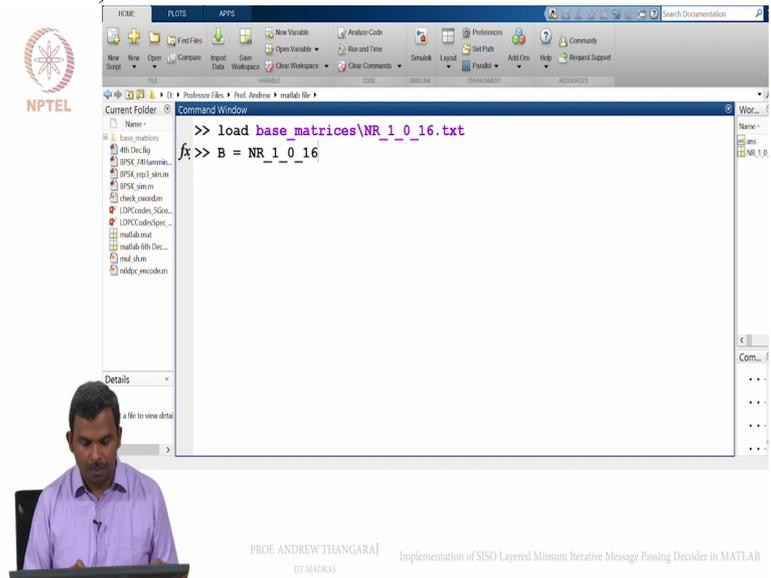
The image shows a MATLAB Command Window with the following text:

```
f>> load base_matrices\NR_1_0_16.txt
```

Below the screenshot, there is a small video inset of Prof. Andrew Thangaraj and a caption: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

So that is the base matrix and for convenience I like to store it in B,

(Refer Slide Time: 01:44)



The image shows a MATLAB Command Window with the following commands and output:

```
>> load base_matrices\NR_1_0_16.txt
fx>> B = NR_1_0_16
```

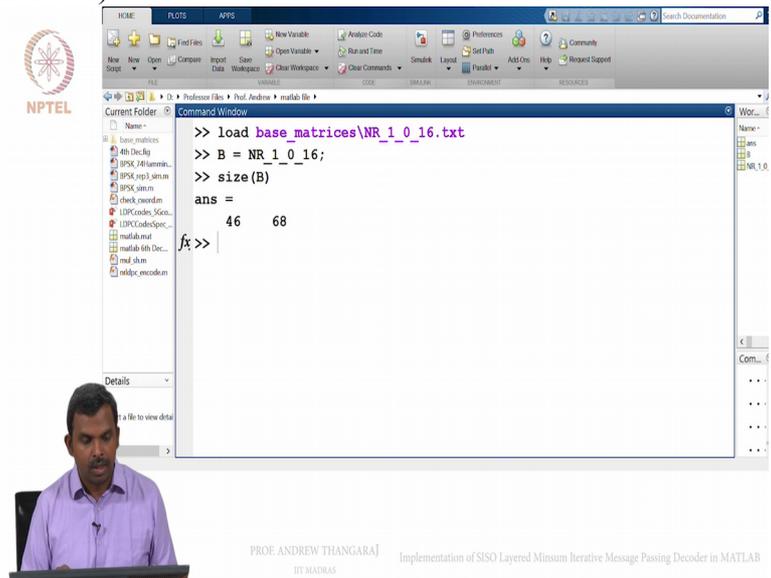
The Command Window also shows a file explorer on the left with the following files:

- base_matrices
- 4th Dec Big
- BPSK_4th Dec
- BPSK_8th Dec
- BPSK_16th Dec
- BPSK_32th Dec
- check_codeword
- LDPCCodesSpec...
- LDPCCodesSpec...
- matlab.mat
- matlab 6th Dec...
- matlab 8th Dec...
- matlab 16th Dec...
- matlab 32th Dec...
- matlab_encoder.m

Below the Command Window, a small video inset shows Prof. Andrew Thangaraj. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok so if you look at size of B,

(Refer Slide Time: 01:47)



The image shows a MATLAB Command Window with the following commands and output:

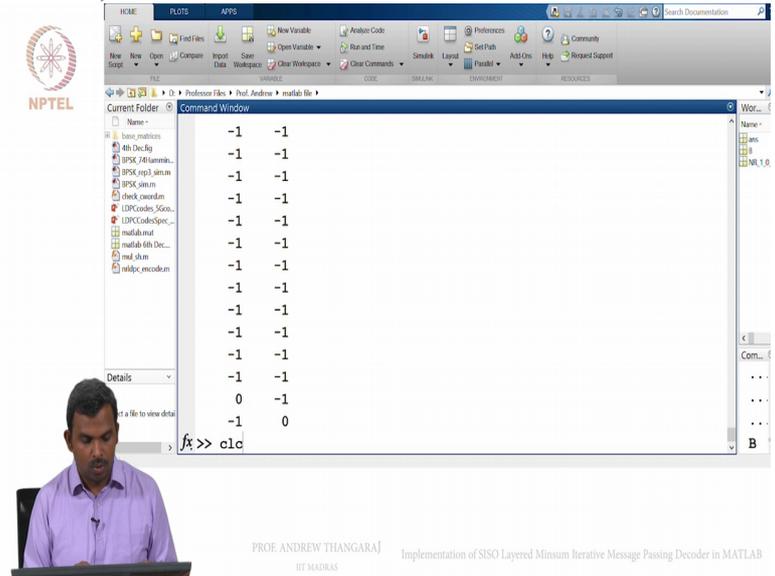
```
>> load base_matrices\NR_1_0_16.txt
>> B = NR_1_0_16;
>> size(B)
ans =
    46    68
fx>>
```

The Command Window also shows the same file explorer as in the previous slide.

Below the Command Window, a small video inset shows Prof. Andrew Thangaraj. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

it is 46 by 68. You remember it is,

(Refer Slide Time: 01:52)



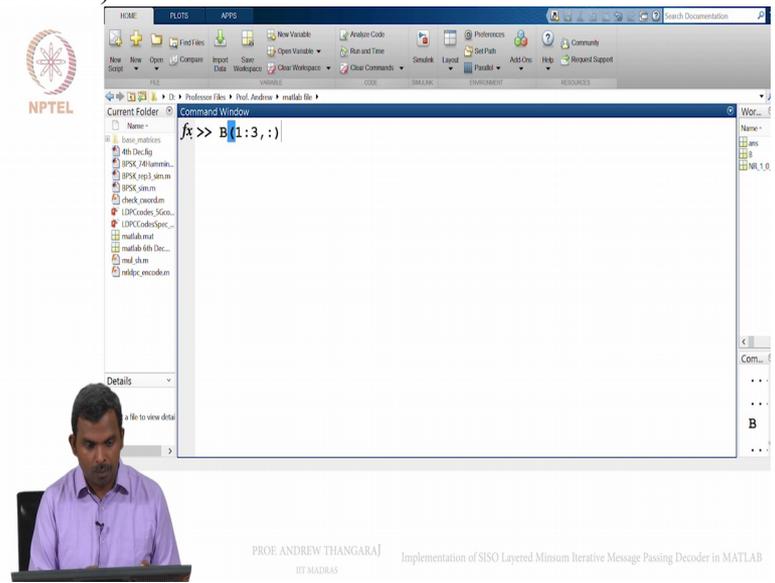
The image shows a MATLAB Command Window with the following matrix displayed:

-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
0	-1
-1	0

Below the window, a small video inset shows Prof. Andrew Thangaraj. The slide footer contains the text: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

B is little bit to look at, may be you can look at the first three or three rows of B,

(Refer Slide Time: 02:03)



The image shows a MATLAB Command Window with the command `B(1:3, :)` entered. Below the window, a small video inset shows Prof. Andrew Thangaraj. The slide footer contains the text: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

(Refer Slide Time: 02:03)

NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok. See whole bunch of minus 1s. So we scroll up a little bit and see the other numbers, Ok. So the

(Refer Slide Time: 02:09)

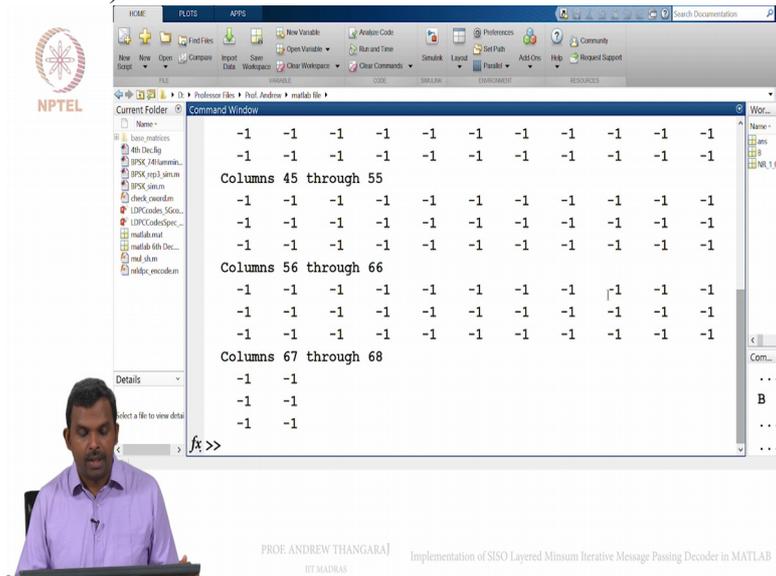
NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

first few columns have some numbers. Remember the expansion is 16 so you will have numbers from minus 1 to 15 in the base matrix. Then you know how this is, right. So if you look at the number 10, you take the 16 by 16 identity matrix and shift it right by 10 positions, Ok. So this is the, this is the base matrix.

(Refer Slide Time: 02:28)



NPTEL

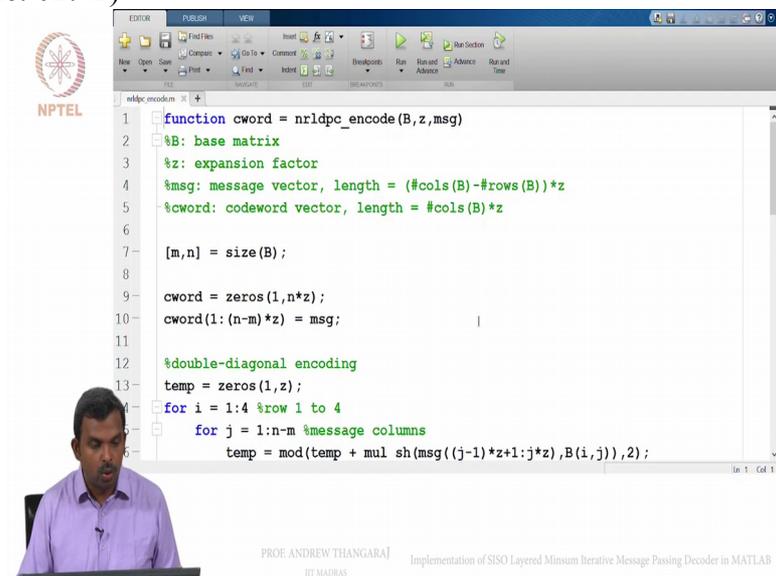
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

And this is expanded to form the parity check matrix, Ok.

So starting with this base matrix we are going to work and we are going to start building a decoder, Ok. So let us move to the MATLAB editing

(Refer Slide Time: 02:42)



NPTEL

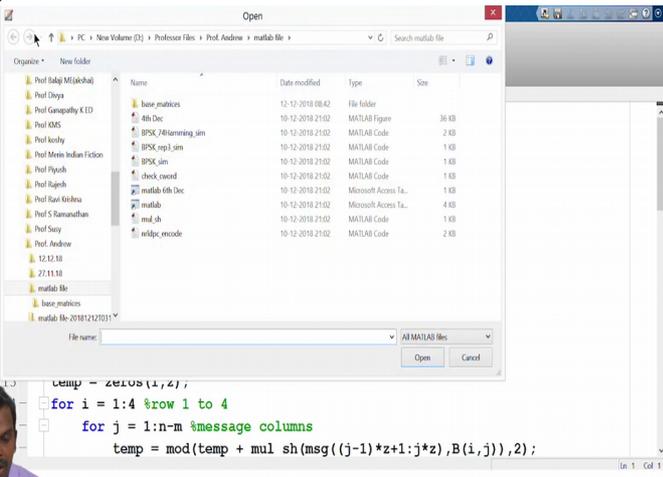
```
1 function cword = nrlrdpc_encode(B,z,msg)
2 %B: base matrix
3 %z: expansion factor
4 %msg: message vector, length = (#cols(B)-#rows(B))*z
5 %cword: codeword vector, length = #cols(B)*z
6
7 [m,n] = size(B);
8
9 cword = zeros(1,n*z);
10 cword(1:(n-m)*z) = msg;
11
12 %double-diagonal encoding
13 temp = zeros(1,z);
14 for i = 1:4 %row 1 to 4
15     for j = 1:n-m %message columns
16         temp = mod(temp + mul sh(msg((j-1)*z+1:j*z),B(i,j)),2);
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

window. This is the encoder, and maybe we need a

(Refer Slide Time: 02:46)



```
temp = zeros(1,4);  
for i = 1:4 %row 1 to 4  
    for j = 1:n-m %message columns  
        temp = mod(temp + mul_sh(msg((j-1)*z+1:j*z),B(i,j)),2);
```

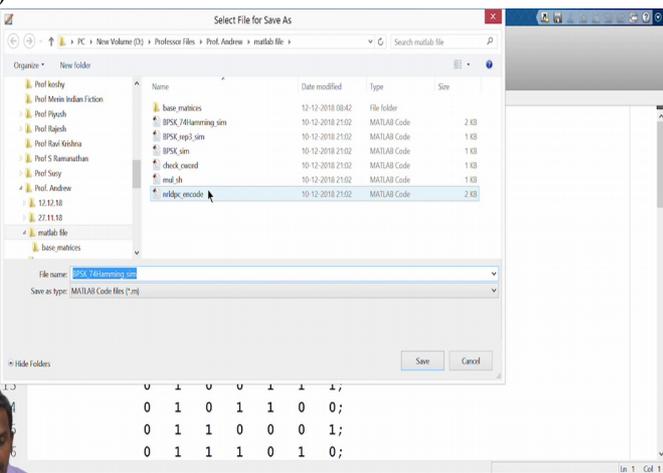
NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

7 4 Hamming code. It is the B P S K 7 4 Hamming code. I am going to save this as

(Refer Slide Time: 02:54)



```
0 1 0 0 1 1 0 0;  
0 1 1 0 0 0 0 1;  
0 1 1 1 0 1 0 0;
```

NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

B P S K n r l d p c sim, Ok.

So we can pick any E_b over N naught. I will pick 4, the rate is sort of difficult to determine. So what we can do here is we can repeat what we did before, so we load, base matrix is, so remember I will run it from my correct directory, base matrices and n r l 0 16, I think it is correct.

(Refer Slide Time: 03:26)



```
EDITOR PUBLISH VIEW
load base_matrices/NR_1_0_16.txt
EbNodB = 4;
R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*R*EbNo));
k = 4; %number of message bits
n = 7; %number of codeword bits
cwords = [0 0 0 0 0 0 0;
           0 0 0 1 0 1 1;
           0 0 1 0 1 1 0;
           0 0 1 1 1 0 1;
           0 1 0 0 1 1 1];
```

Let me check that once again.

So that is the command we ran, 1 0 16, right so that is the thing. So that is Ok. We loaded that, Ok and then we set B to be equal to 1 0 16. So now I will have my

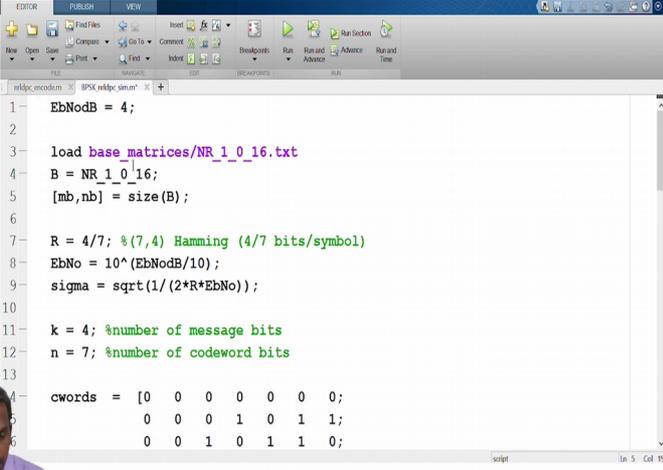
(Refer Slide Time: 03:46)



```
EDITOR PUBLISH VIEW
load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
EbNodB = 4;
R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*R*EbNo));
k = 4; %number of message bits
n = 7; %number of codeword bits
cwords = [0 0 0 0 0 0 0;
           0 0 0 1 0 1 1;
           0 0 1 0 1 1 0;
           0 0 1 1 1 0 1];
```

base matrix in this B. And now I am ready to find out what m and n are. So I will have, need the size of B. I will use,

(Refer Slide Time: 04:02)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6
7 R = 4/7; % (7,4) Hamming (4/7 bits/symbol)
8 EbNo = 10^(EbNodB/10);
9 sigma = sqrt(1/(2*R*EbNo));
10
11 k = 4; %number of message bits
12 n = 7; %number of codeword bits
13
14 cwords = [0 0 0 0 0 0 0;
15           0 0 0 1 0 1 1;
16           0 0 1 0 1 1 0];
```

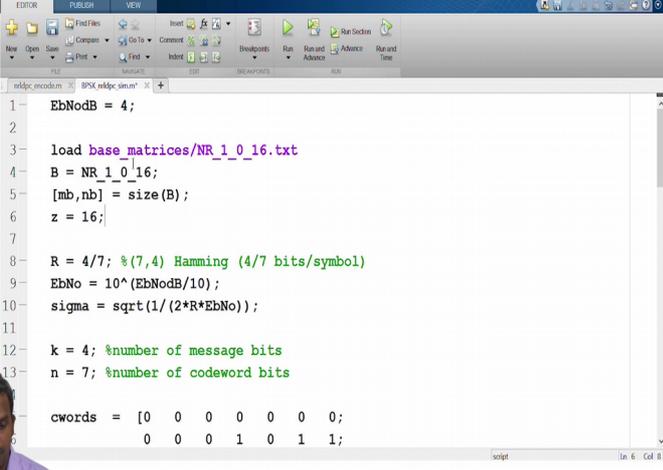
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

I will use this.

So we know what m b is going to be, right. m b is going to be 46. n b is going to be 68, right. So that is the N R 1 0 16. If I change it to something else, this will change, Ok. And also clearly z which is the expansion factor

(Refer Slide Time: 04:20)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 R = 4/7; % (7,4) Hamming (4/7 bits/symbol)
9 EbNo = 10^(EbNodB/10);
10 sigma = sqrt(1/(2*R*EbNo));
11
12 k = 4; %number of message bits
13 n = 7; %number of codeword bits
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1;
17           0 0 1 0 1 1 0];
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

is 16, Ok.

So once I do this I will be able to determine these things.

(Refer Slide Time: 04:29)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
9 EbNo = 10^(EbNodB/10);
10 sigma = sqrt(1/(2*R*EbNo));
11
12 cwords = [0 0 0 0 0 0 0;
13           0 0 0 1 0 1 1;
14           0 0 1 0 1 1 0;
15           0 0 1 1 1 0 1;
16           0 1 0 0 1 1 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Let me put this little bit forward. k equals

(Refer Slide Time: 04:33)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 k = 4; %number of message bits
9 n = 7; %number of codeword bits
10
11 R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

n b minus m b times z. These are the number of message bits.

(Refer Slide Time: 04:40)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 k = (nb-mb)*z; %number of message bits
9 n = 7; %number of codeword bits
10
11 R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
12 EbNo = 10*(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1];
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

And the number of codeword bits, so usually when you generate the codeword, I am going to generate the entire codeword but and we will simulate that in this code. But typically in, in wireless standards people do something called the rate matching.

So you will do some puncturing, you will do some shortening, combination of these things and put it into some circular buffer, read from that etc. So all of that we are not going to do with this, in this lecture or this class. We will just assume the whole base matrix and the whole parity check matrix is what we simulate, Ok.

So later on, maybe if we have time, I will go through some examples where may be I will show you some puncturing but it is quite easy to program the puncturing part. At least I will mention how that is accommodated, Ok.

So n as far as I am concerned, is n b times z.

(Refer Slide Time: 05:29)



```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 k = (nb-mb)*z; %number of message bits
9 n = nb*z; %number of codeword bits
10
11 R = 4/7; %(7,4) Hamming (4/7 bits/symbol)
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1];
```

PROF. ANDREW THANGARAJ
IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok. So remember once again, this is, this is, n b is 46, m b is 68. So the difference is 22. So k will be 22 into 16. And n will be 68 into 16. So that is the current thing I am working, Ok. So the rate is going to be k by n.

(Refer Slide Time: 05:52)

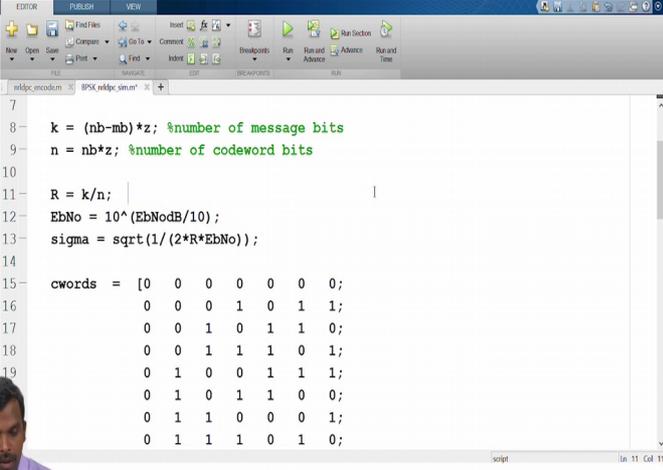


```
1 EbNodB = 4;
2
3 load base_matrices/NR_1_0_16.txt
4 B = NR_1_0_16;
5 [mb,nb] = size(B);
6 z = 16;
7
8 k = (nb-mb)*z; %number of message bits
9 n = nb*z; %number of codeword bits
10
11 R = k/n; %(7,4) Hamming (4/7 bits/symbol)
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1];
```

PROF. ANDREW THANGARAJ
IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So this is clearly not the Hamming code. So, oops, oops, it is going all over the place. It is not deleted but Ok. Let us just do it. Ok.

(Refer Slide Time: 06:08)



```
7
8 k = (nb-mb)*z; %number of message bits
9 n = nb*z; %number of codeword bits
10
11 R = k/n;
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 cwords = [0 0 0 0 0 0 0;
16           0 0 0 1 0 1 1;
17           0 0 1 0 1 1 0;
18           0 0 1 1 1 0 1;
19           0 1 0 0 1 1 1;
20           0 1 0 1 1 0 0;
21           0 1 1 0 0 0 1;
22           0 1 1 1 0 1 0;
23           1 0 0 0 1 0 1;
24           1 0 0 1 1 1 0;
25           1 0 1 0 0 1 1;
26           1 0 1 1 0 0 0;
27           1 1 0 0 0 1 0;
28           1 1 0 1 0 0 1;
29           1 1 1 0 1 0 0;
30           1 1 1 1 1 1 1];
script lin 11 Col 11
```

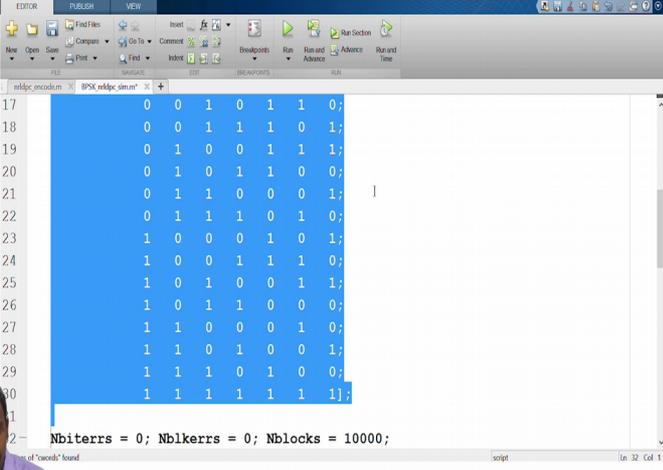


PROF. ANDREW THANGARAJ
IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So once you do this, you have E_b/N_0 naught, you have a base matrix, you have the rate and so you can find your sigma, Ok.

So there is no way I can list out all the codewords. So that part is completely

(Refer Slide Time: 06:19)



```
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532

```

(Refer Slide Time: 06:20)



```
EDITOR  PUBLISH  VIEW
+-----+-----+-----+
| Find Files | Go To | Comment | Breakpoints | Run and Advance | Run and Time |
| New | Open | Save | Print | Find | Insert | Run | Run and Advance | Run and Time |
+-----+-----+-----+
nbpdc_encoder.m  | BPSK_nbpc_sim.m  |
7
8   k = (nb-mb)*z; %number of message bits
9   n = nb*z; %number of codeword bits
10
11  R = k/n;
12  EbNo = 10^(EbNodB/10);
13  sigma = sqrt(1/(2*R*EbNo));
14
15
16  Nbiterrs = 0; Nblkerrs = 0; Nblocks = 10000;
17  for i = 1: Nblocks
18      msg = randi([0 1],1,k); %generate random k-bit message
19      %Encoding
20      cword = [msg mod(msg(1)+msg(2)+msg(3) ,2) ...
21              mod(msg(2)+msg(3)+msg(4) ,2) ...
22              mod(msg(1)+msg(2)+msg(4) ,2)]; % (7,4) Hamming
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
script  ln 15 Col 1
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So the other parts of the simulation remain the same. You generate the k-bit random sequence and then

(Refer Slide Time: 06:27)



```
EDITOR  PUBLISH  VIEW
+-----+-----+-----+
| Find Files | Go To | Comment | Breakpoints | Run and Advance | Run and Time |
| New | Open | Save | Print | Find | Insert | Run | Run and Advance | Run and Time |
+-----+-----+-----+
nbpdc_encoder.m  | BPSK_nbpc_sim.m  |
12  EbNo = 10^(EbNodB/10);
13  sigma = sqrt(1/(2*R*EbNo));
14
15  Nbiterrs = 0; Nblkerrs = 0; Nblocks = 10000;
16  for i = 1: Nblocks
17      msg = randi([0 1],1,k); %generate random k-bit message
18      %Encoding
19      cword = [msg mod(msg(1)+msg(2)+msg(3) ,2) ...
20              mod(msg(2)+msg(3)+msg(4) ,2) ...
21              mod(msg(1)+msg(2)+msg(4) ,2)]; % (7,4) Hamming
22
23      s = 1 - 2 * cword; %BPSK bit to symbol conversion
24      r = s + sigma * randn(1,n); %AWGN channel I
25
26      %Hard-decision decoding
27      b = (r < 0); %threshold at zero
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
script  ln 26 Col 1
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

you start encoding.

So now remember I want to focus on the decoder. I do not want to be stuck with, stuck with; I am just testing my decoder in some sense. So I will not be doing a lot of blocks. I will just do one block and I do not really want to be encoding. So I will just comment this out.

I am not going to be encoding. And what I will do instead is I will simply assume the messages 0s. Ok so I

(Refer Slide Time: 06:55)



```
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
16 for i = 1: Nblocks
17     %msg = randi([0 1],1,k); %generate random k-bit message
18     msg = zeros(1,k);
19     %Encoding
20     cword = [msg mod(msg(1)+msg(2)+msg(3) ,2) ...
21             mod(msg(2)+msg(3)+msg(4) ,2) ...
22             mod(msg(1)+msg(2)+msg(4) ,2)]; % (7,4) Hamming
23
24     s = 1 - 2 * cword; %BPSK bit to symbol conversion
25     r = s + sigma * randn(1,n); %AWGN channel I
26
27     %Hard-decision decoding
```

PROF. ANDREW THANGARA
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

will transfer the all-zero codeword always, Ok all-zero message and all-zero codeword always.

It turns out that is enough for decoding checking of course and the decoder I will not assume all-zero codeword is being transmitted. I will try to decode it based on the received codeword.

And this way I bypass the encoding requirement, Ok. We know we have an encoder input program wherein it just adds the complications. We can do that later on. For now onwards assume all-zero codeword, Ok.

(Refer Slide Time: 07:23)

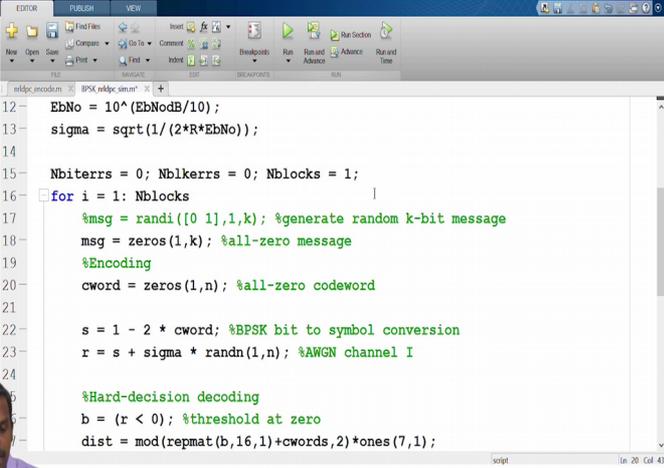


```
12 EbNo = 10^(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
16 for i = 1: Nblocks
17     %msg = randi([0 1],1,k); %generate random k-bit message
18     msg = zeros(1,k); %all-zero message
19     %Encoding
20     cword = [msg mod(msg(1)+msg(2)+msg(3) ,2) ...
21             mod(msg(2)+msg(3)+msg(4) ,2) ...
22             mod(msg(1)+msg(2)+msg(4) ,2)]; % (7,4) Hamming
23
24     s = 1 - 2 * cword; %BPSK bit to symbol conversion
25     r = s + sigma * randn(1,n); %AWGN channel I
26
27     %Hard-decision decoding
```

PROF. ANDREW THANGARA
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So you know, all of us know when all-zero codeword is encoded you get, all-zero messages encoded with any linear code, you are going to get the all-zero codeword, Ok.

(Refer Slide Time: 07:40)



PROF. ANDREW THANGARAJ
IIT MADRAS

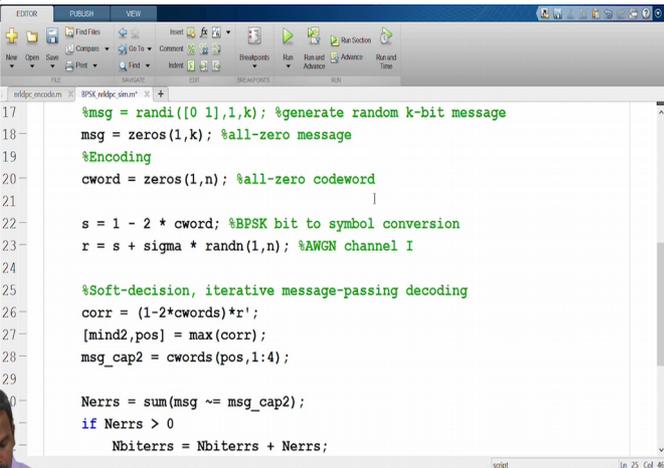
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

```
12 EbNo = 10*(EbNodB/10);
13 sigma = sqrt(1/(2*R*EbNo));
14
15 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
16 for i = 1: Nblocks
17     %msg = randi([0 1],1,k); %generate random k-bit message
18     msg = zeros(1,k); %all-zero message
19     %Encoding
20     cword = zeros(1,n); %all-zero codeword
21
22     s = 1 - 2 * cword; %BPSK bit to symbol conversion
23     r = s + sigma * randn(1,n); %AWGN channel I
24
25     %Hard-decision decoding
26     b = (r < 0); %threshold at zero
27     dist = mod(repmat(b,16,1)+cwords,2)*ones(7,1);
```

So this is something we do for simplicity in the decoder.

Ok so after that things are Ok. It is exactly the same, nothing much to change. We are not going to be doing hard decision decoding. We are going to be doing soft decision decoding. In fact this is going to be soft decision iterative message passing decoding.

(Refer Slide Time: 08:04)



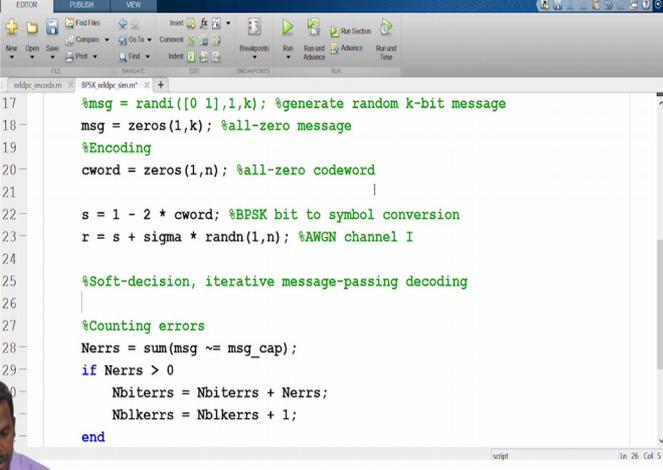
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

```
17 %msg = randi([0 1],1,k); %generate random k-bit message
18 msg = zeros(1,k); %all-zero message
19 %Encoding
20 cword = zeros(1,n); %all-zero codeword
21
22 s = 1 - 2 * cword; %BPSK bit to symbol conversion
23 r = s + sigma * randn(1,n); %AWGN channel I
24
25 %Soft-decision, iterative message-passing decoding
26 corr = (1-2*cwords)*r';
27 [mind2,pos] = max(corr);
28 msg_cap2 = cwords(pos,1:4);
29
30 Nerrs = sum(msg ~= msg_cap2);
31 if Nerrs > 0
32     Nbiterrs = Nbiterrs + Nerrs;
```

So all of these things do not apply. You will have only one message cap. So this should be good enough. This is counting errors, right, so let me put that as counting errors.

(Refer Slide Time: 08:19)



```
17 %msg = randi([0 1],1,k); %generate random k-bit message
18 msg = zeros(1,k); %all-zero message
19 %Encoding
20 cword = zeros(1,n); %all-zero codeword
21
22 s = 1 - 2 * cword; %BPSK bit to symbol conversion
23 r = s + sigma * randn(1,n); %AWGN channel I
24
25 %Soft-decision, iterative message-passing decoding
26
27 %Counting errors
28 Nerrs = sum(msg ~= msg_cap);
29 if Nerrs > 0
30     Nbiterrs = Nbiterrs + Nerrs;
31     Nblkerrs = Nblkerrs + 1;
32 end
```



PROF. ANDREW THANGARAJ
IIT MADRAS

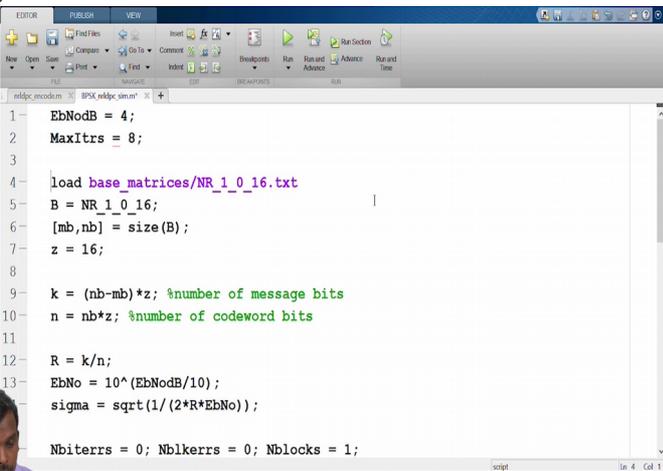
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So now all that remains is the decoder. Ok so hopefully we have seen I have got the setting completely done, everything else is fine, I only need to figure out the decoder, Ok.

So I have loaded my base matrix, I have loaded my expansion factor, I have encoded and I have my received vector and I have to do my decoding. So all I to do is decoding, Ok. It is simple enough.

So in the decoder we know it is iterative so we need to figure number of iterations. So maybe that is the parameter you want to set. We could it here. Max iterations equals, I know 8, does not matter

(Refer Slide Time: 08:57)



```
1 EbNodB = 4;
2 MaxItrs = 8;
3
4 load base_matrices/NR_1_0_16.txt
5 B = NR_1_0_16;
6 [mb,nb] = size(B);
7 z = 16;
8
9 k = (nb-mb)*z; %number of message bits
10 n = nb*z; %number of codeword bits
11
12 R = k/n;
13 EbNo = 10^(EbNodB/10);
14 sigma = sqrt(1/(2*R*EbNo));
15
16 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
```



PROF. ANDREW THANGARAJ
IIT MADRAS

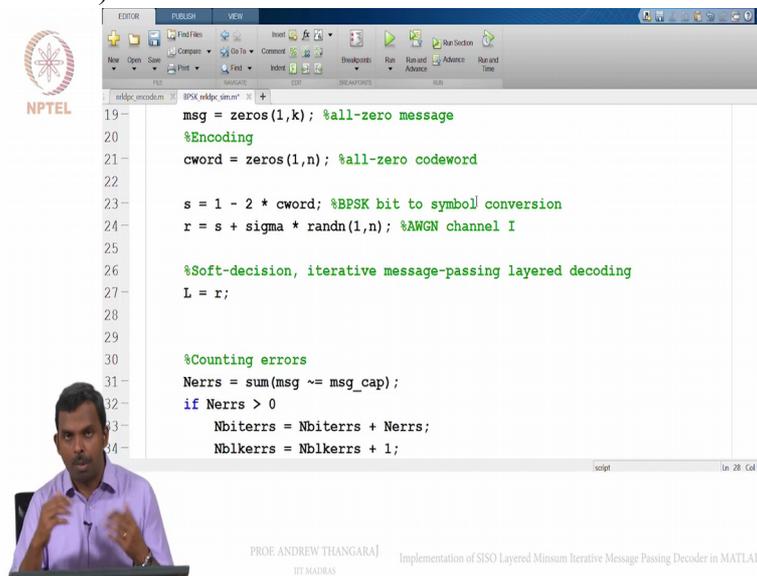
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

what we put here.

So, so that is the max iterations, Ok. We will keep that. We will use it, when we have to. So now the iterative message passing decoding, if you remember we are going to do layer decoding. So let me, layer I will add that, Ok.

If you remember layer decoding we have one total belief vector which is initialized to r in the beginning, Ok so that is my L . Ok

(Refer Slide Time: 09:26)



The screenshot shows a MATLAB script editor with the following code:

```
19 msg = zeros(1,k); %all-zero message
20 %Encoding
21 cword = zeros(1,n); %all-zero codeword
22
23 s = 1 - 2 * cword; %BPSK bit to symbol conversion
24 r = s + sigma * randn(1,n); %AWGN channel I
25
26 %Soft-decision, iterative message-passing layered decoding
27 L = r;
28
29
30 %Counting errors
31 Nerrs = sum(msg ~= msg_cap);
32 if Nerrs > 0
33     Nbiterrs = Nbiterrs + Nerrs;
34     Nblkerrs = Nblkerrs + 1;
```

Below the code, there is a small video inset of Prof. Andrew Thangaraj, a man in a light blue shirt, speaking. The NPTEL logo is visible in the top left corner of the slide. The bottom of the slide contains the text: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

you can go back and look at the slides if you like to see where this notation is coming from.

I did a toy example with L being the total belief, Ok so now that is, that is set as r which is the received vector. Now I am going to start working with r , right. So you remember I am going to process one layer at a time and for me one layer will be one block row in the base matrix.

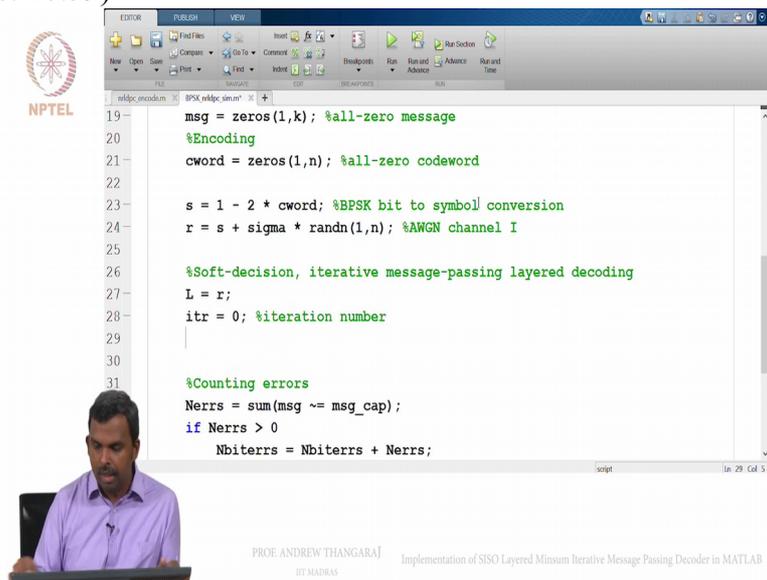
So you remember my base matrix has 46 rows. I am going to take the first row and process it. Ok. I am going to treat my first row as my first layer, Ok. So I will process that and then add to the total belief and proceed, Ok. So that is what I am going to do first, Ok.

So L equals r and then I have to start doing the first row. So how do I do the first row? So if you remember B is my base matrix, it has a bunch of minus 1s, Ok. And those minus 1s are not very relevant for me. I just have to go through and find out where the non minus 1 values are, Ok and then based on that do my row processing, right.

So that is, that is how it works. So it is, you can, you can do a lot of efficiency into this. Once again I am not going to worry too much about the efficiency. So the outer loop, the outer loop will be the number of iterations.

So I will have itr be equal to 0. This is the iteration number, Ok.

(Refer Slide Time: 10:59)



The screenshot displays a MATLAB script with the following code:

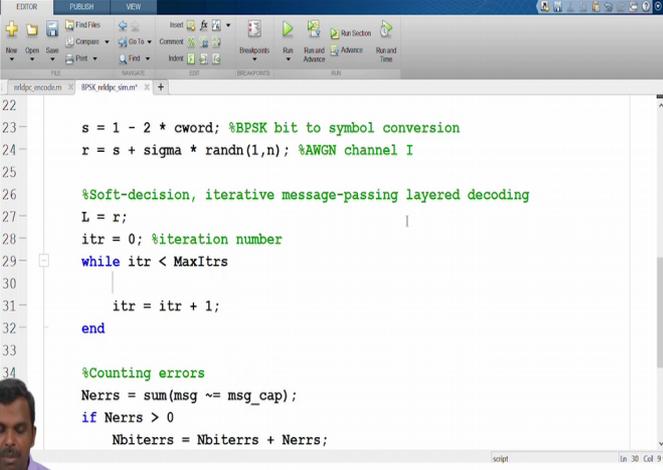
```
19 msg = zeros(1,k); %all-zero message
20 %Encoding
21 cword = zeros(1,n); %all-zero codeword
22
23 s = 1 - 2 * cword; %BPSK bit to symbol conversion
24 r = s + sigma * randn(1,n); %AWGN channel I
25
26 %Soft-decision, iterative message-passing layered decoding
27 L = r;
28 itr = 0; %iteration number
29
30
31 %Counting errors
Nerrs = sum(msg ~= msg_cap);
if Nerrs > 0
    Nbiterrs = Nbiterrs + Nerrs;
```

Below the code, a small video inset shows Prof. Andrew Thangaraj, an IT Madras professor, sitting at a desk. The text below the video reads: "PROF. ANDREW THANGARAJ, ITT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

And I will also have, so there are various ways to do this thing, so, so one can put a loop here on while itr as less than max iterations, right and end, Ok then over all I will have itr equals itr plus 1, Ok.

So you may

(Refer Slide Time: 11:25)



```
22
23 s = 1 - 2 * cword; %BPSK bit to symbol conversion
24 r = s + sigma * randn(1,n); %AWGN channel I
25
26 %Soft-decision, iterative message-passing layered decoding
27 L = r;
28 itr = 0; %iteration number
29 while itr < MaxItrs
30     |
31     itr = itr + 1;
32 end
33
34 %Counting errors
Nerrs = sum(msg ~= msg_cap);
if Nerrs > 0
    Nbiterrs = Nbiterrs + Nerrs;
```



PROF. ANDREW THANGARA
IIT MADRAS

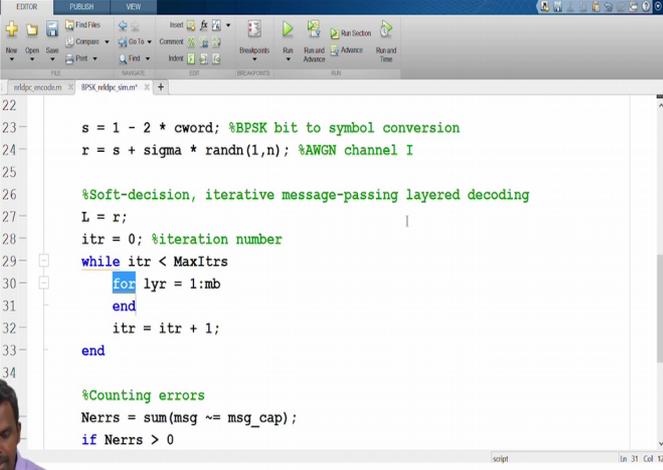
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

me why I am not doing a for loop. So it turns out I want to terminate in the middle if I have valid code word as the answer. So that is why I like a while loop here. So this is my loop for the iterations, Ok.

So I will repeat for every iteration a certain, the row processing steps, right, process all the rows. So I put the loop for the iterations, Ok. And inside this iteration I am going to process the parity check matrix and multiple layers here. And each layer I will need a loop as well, Ok.

So that is going to be, I will call it layer equals 1 colon m b, right. So that is my number of layers

(Refer Slide Time: 12:05)



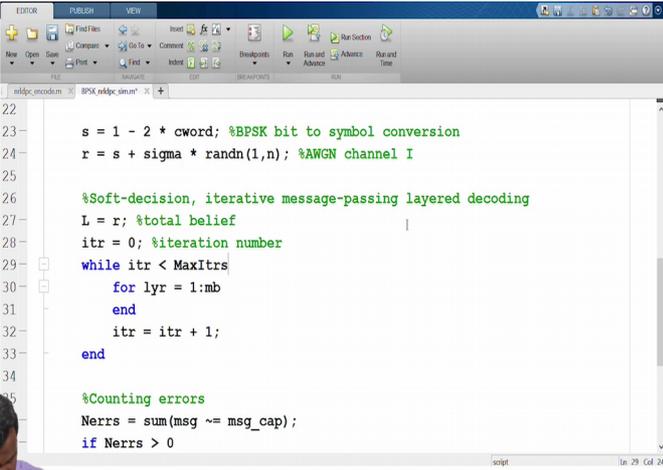
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

```
22
23 s = 1 - 2 * cword; %BPSK bit to symbol conversion
24 r = s + sigma * randn(1,n); %AWGN channel I
25
26 %Soft-decision, iterative message-passing layered decoding
27 L = r;
28 itr = 0; %iteration number
29 while itr < MaxItrs
30     for lyr = 1:mb
31     end
32     itr = itr + 1;
33 end
34
35 %Counting errors
36 Nerrs = sum(msg ~= msg_cap);
37 if Nerrs > 0
```

PROF. ANDREW THANGARAJ
IIT MADRAS

and that ends there, Ok. Once again what is L? L is my total belief, Ok.

(Refer Slide Time: 12:13)



Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

```
22
23 s = 1 - 2 * cword; %BPSK bit to symbol conversion
24 r = s + sigma * randn(1,n); %AWGN channel I
25
26 %Soft-decision, iterative message-passing layered decoding
27 L = r; %total belief
28 itr = 0; %iteration number
29 while itr < MaxItrs
30     for lyr = 1:mb
31     end
32     itr = itr + 1;
33 end
34
35 %Counting errors
36 Nerrs = sum(msg ~= msg_cap);
37 if Nerrs > 0
```

PROF. ANDREW THANGARAJ
IIT MADRAS

So lot of people call belief as log likelihood ratio, I am just using a slightly more generic terminology here.

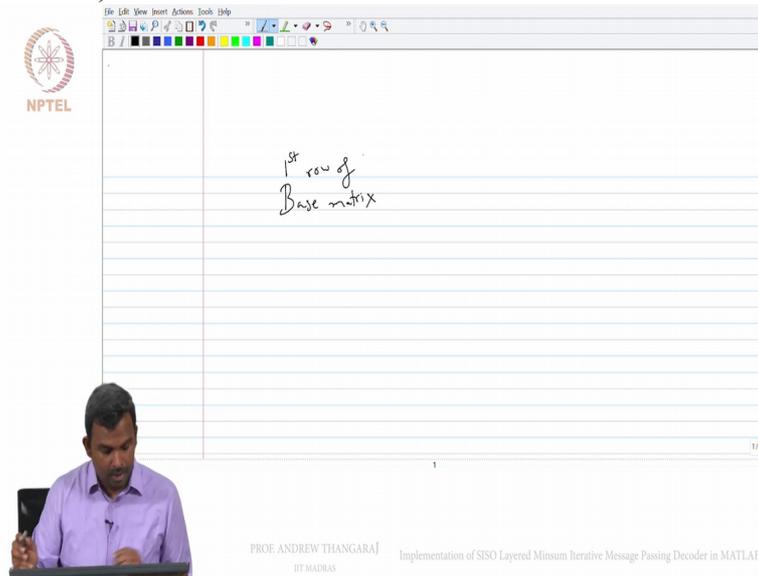
If somebody says L is log likelihood ratio, technically it is proportional to log likelihood ratio. There is 2 by σ^2 factor there. So that is why I am calling belief.

You can use $L = r$ as well, I mean that is not wrong. That is quite reasonable to say, Ok. And what do we do in each layer? In each layer I have to do row processing, Ok. So it turns out,

most people like to do this in parallel. So one can do that as well. And we will do some sort of parallel effort here.

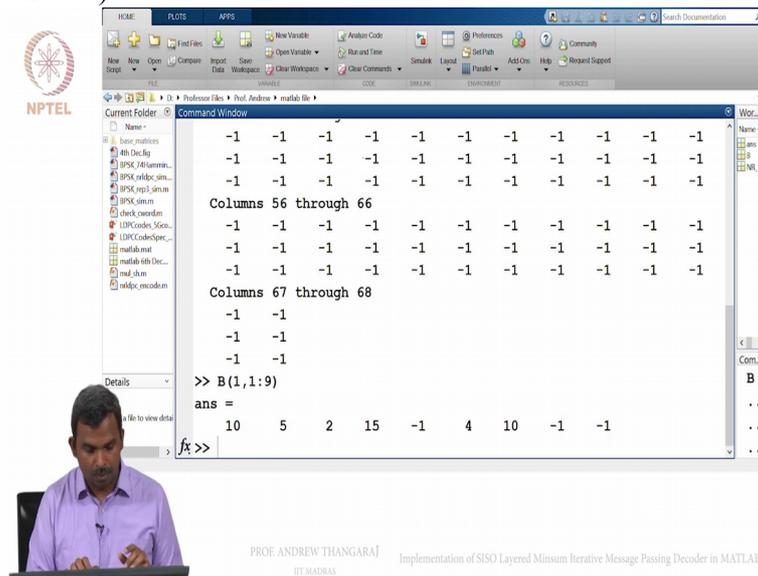
So, so remember what is going to be each layer? So maybe I should take out this thing? So if we look at each layer, so, so the base matrix B, base matrix, it is basically the first row of base matrix,

(Refer Slide Time: 13:21)



we saw that here (()) 13:25 B of 1 comma 1 colon 9

(Refer Slide Time: 13:30)



or something, so that comes in the first row.

So you see the first row is 10, 5, 2, 15, minus 1, 4, 10, minus 1, minus 1. There are lots of minus 1s but there is 10, 5 etc. So let me just copy this over to inside, so...I do not need to be exact. Just, just for rough purposes, this is 10, 5, 2, 15, minus 1 Ok. So you had 10, 5, 2, 15, minus 1.

Remember what is 10? 10 is 16 by 16

(Refer Slide Time: 14:03)

The slide shows a whiteboard with the following handwritten text: "1st row of Base matrix [10 5 2 15 -1". The slide also features the NPTEL logo in the top left corner and a small video inset of Prof. Andrew Thangaraj in the bottom left. At the bottom, the text reads "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

identity shifted right by 10 columns right? So rotate shifted

(Refer Slide Time: 14:17)

The slide shows a whiteboard with the following handwritten text: "1st row of Base matrix [10 5 2 15 -1" followed by a downward arrow and "16x16 identity shifted right by 10 columns". The slide also features the NPTEL logo in the top left corner and a small video inset of Prof. Andrew Thangaraj in the bottom left. At the bottom, the text reads "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

right by 10 columns. So it will have something like this, right.

10, that would move here Ok something like that. So this would be this block and then the 5

(Refer Slide Time: 14:31)

The slide shows a handwritten matrix on a whiteboard. The matrix is:

$$\begin{bmatrix} 10 & 5 & 2 & 15 & -1 \end{bmatrix}$$

Handwritten notes include:

- "1st row of Base matrix" with an arrow pointing to the first row of the matrix.
- "16x16 identity shifted right by 10 columns" with an arrow pointing to the right side of the matrix.

The presenter, Prof. Andrew Thangaraj, is visible in the bottom left corner. The slide title is "Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

would come. 5 would be probably longer here and then shorter here, Ok and then 2 etc,

(Refer Slide Time: 14:38)

The slide shows a handwritten matrix on a whiteboard. The matrix is:

$$\begin{bmatrix} 10 & 5 & 2 & 15 & -1 \end{bmatrix}$$

Handwritten notes include:

- "1st row of Base matrix" with an arrow pointing to the first row of the matrix.
- "16x16 identity shifted right by 10 columns" with an arrow pointing to the right side of the matrix.

The presenter, Prof. Andrew Thangaraj, is visible in the bottom left corner. The slide title is "Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok. So the parity check matrix is expanded from here. And there is some shift going on here, Ok. That is number 1.

Number 2, if you remember every entry in the base matrix I need a storage matrix,

(Refer Slide Time: 14:57)

1st row of Base matrix

Storage matrix

16x16 identity shifted right by 10 columns

10 5 2 15 -1

NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

right. So that is the message that is being exchanged. So that storage matrix actually needs for the expanded version, right? So I need a fairly big storage matrix to work with. Ok without the storage matrix things are not going to work.

Ok so that is the next big thing that we will define carefully, Ok. So how will we define the storage matrix? It needs to go outside of, outside of this you define a storage matrix based on B, Ok. And how is the storage matrix going to work?

And remember I do not need a storage for every entry in B. B has a lot of entries, 46 by 68 but quite a few of them are minus 1s, Ok.

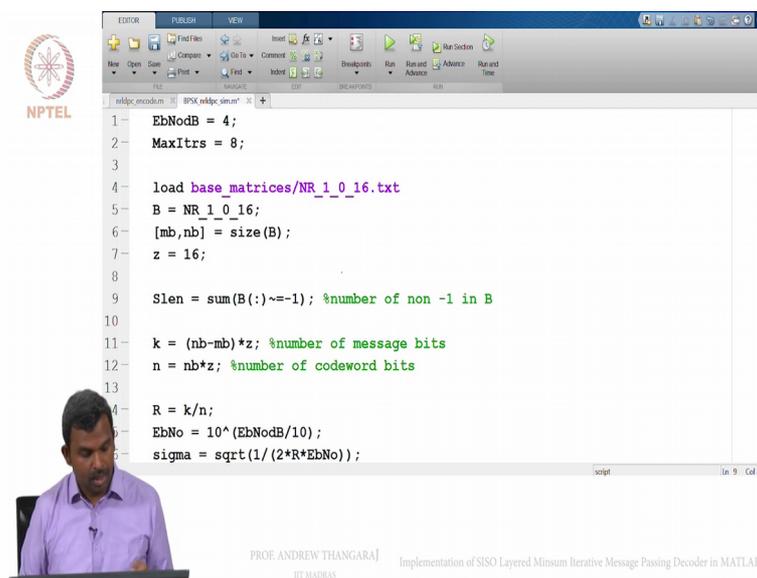
For every minus 1, I do not need to store anything. Only if it is not minus 1, I need to store something. And how much do I need to store? For every entry, non-minus 1 entry of B, I need to store 16 values, right. That is the expansion by 16, right. That is only identity, Ok.

So 16 1s are there for every entry in the base matrix or more generally, z 1s are there for every entry in the base matrix, Ok. So I will find out how many non-minus 1 entries are there, where they are and, and sort of where they are in this will give me and wherever I have a non-minus 1, I need to store 16 values, Ok.

So I will store 16 values in a sequence and then we will have to do some rotations. We will come to that. We will deal with this but for now let us declare this storage array. Ok that is quite important. We need to do that, Ok.

So how do we do this? So, so first I would need to find number of non-minus 1s. So I will call it S len equals length of, or well you can even do this, sum of B of colon not equal to minus 1, Ok. So this is going to give you a total count of number of non-minus 1 in B, Ok.

(Refer Slide Time: 17:08)



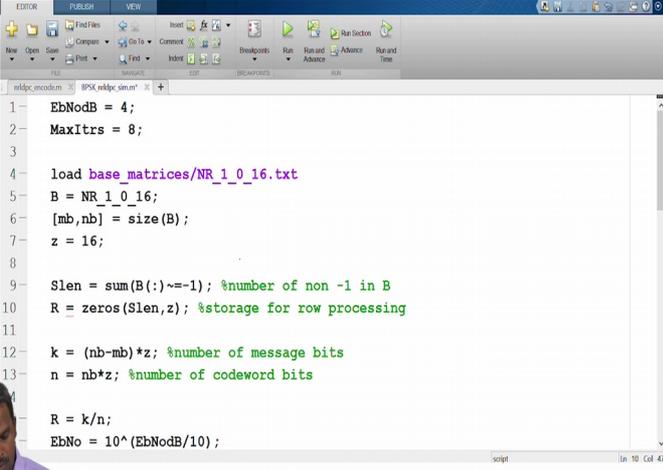
```
1 EbNodB = 4;
2 MaxItrs = 8;
3
4 load base_matrices/NR_1_0_16.txt
5 B = NR_1_0_16;
6 [mb,nb] = size(B);
7 z = 16;
8
9 Slen = sum(B(:)~= -1); %number of non -1 in B
10
11 k = (nb-mb)*z; %number of message bits
12 n = nb*z; %number of codeword bits
13
14 R = k/n;
15 EbNo = 10^(EbNodB/10);
16 sigma = sqrt(1/(2*R*EbNo));
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok.

So why did I do colon? I just wanted to make into one vector, so that I can do this quite reasonably. So you can, you can get back and then once you have S len, you can define your storage, Ok. So my storage, I am going to call it R, R is zeros for now of S len comma z, this is the storage for row processing.

(Refer Slide Time: 17:41)

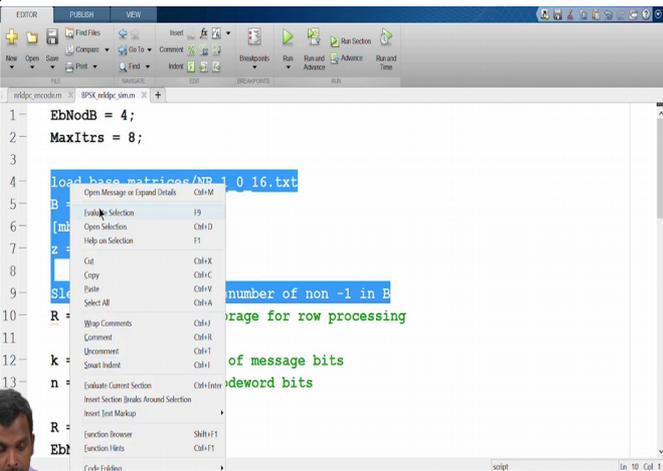


```
1 EbNodB = 4;  
2 MaxItrs = 8;  
3  
4 load base_matrices/NR_1_0_16.txt  
5 B = NR_1_0_16;  
6 [mb,nb] = size(B);  
7 z = 16;  
8  
9 Slen = sum(B(:)~= -1); %number of non -1 in B  
10 R = zeros(Slen,z); %storage for row processing  
11  
12 k = (nb-mb)*z; %number of message bits  
13 n = nb*z; %number of codeword bits  
14  
15 R = k/n;  
16 EbNo = 10^(EbNodB/10);
```

PROF. ANDREW THANGARAJ
IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok, so hopefully you agree that this is correct. So maybe we can run up to this and see what the value is,

(Refer Slide Time: 17:49)

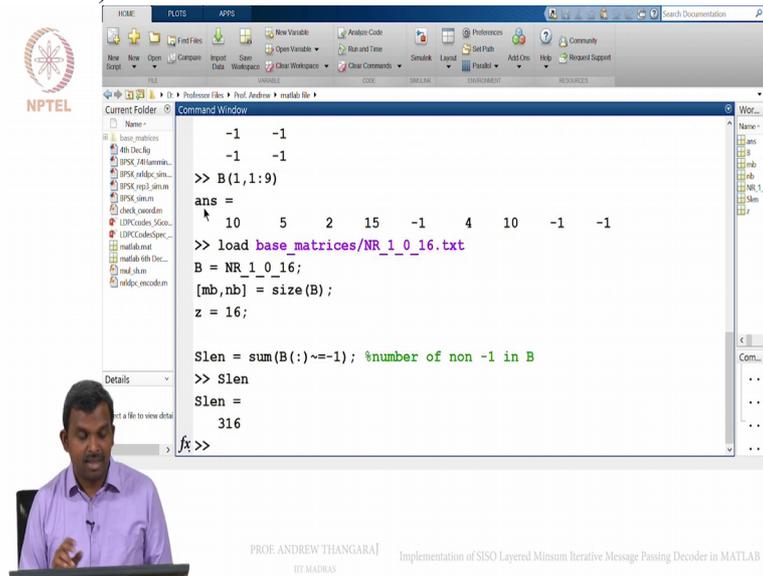


```
1 EbNodB = 4;  
2 MaxItrs = 8;  
3  
4 load base_matrices/NR_1_0_16.txt  
5 B = NR_1_0_16;  
6 [mb,nb] = size(B);  
7 z = 16;  
8 Slen = sum(B(:)~= -1); %number of non -1 in B  
9 R = zeros(Slen,z); %storage for row processing  
10  
11 k = (nb-mb)*z; %number of message bits  
12 n = nb*z; %number of codeword bits  
13  
14 R = k/n;  
15 EbNo = 10^(EbNodB/10);
```

PROF. ANDREW THANGARAJ
IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

evaluate selection. If you do that you will see that S len is. It is

(Refer Slide Time: 17:57)



The screenshot shows the MATLAB Command Window with the following code and output:

```
-1 -1
-1 -1
>> B(1,1:9)
ans =
    10     5     2    15    -1     4    10    -1    -1
>> load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

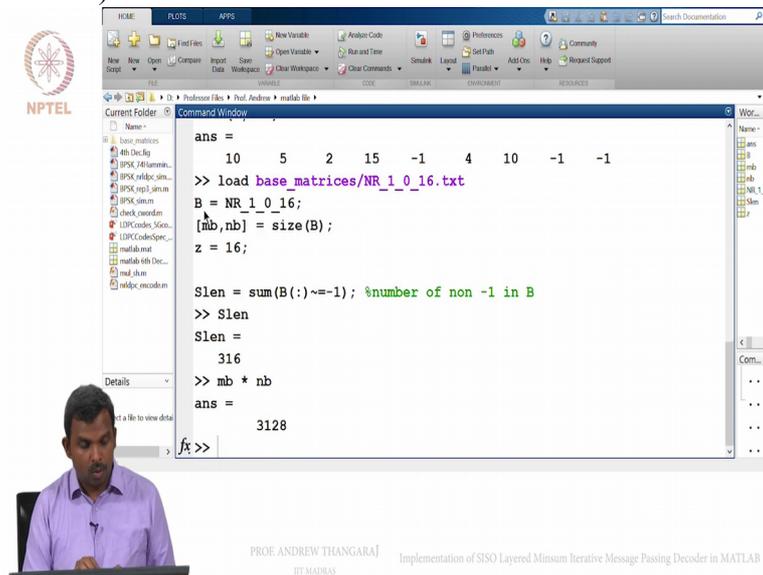
Slen = sum(B(:)~= -1); %number of non -1 in B
>> Slen
Slen =
    316
fx>>
```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, an IIT Madras professor, speaking. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

316, Ok.

So even though there are, so if you remember, B is, if you do m b times n b

(Refer Slide Time: 18:05)



The screenshot shows the MATLAB Command Window with the following code and output:

```
ans =
    10     5     2    15    -1     4    10    -1    -1
>> load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~= -1); %number of non -1 in B
>> Slen
Slen =
    316
>> mb * nb
ans =
    3128
fx>>
```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, an IIT Madras professor, speaking. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

you get 3128, so that is the size of, total number of entries in B is 3128, 46 by 68 but out of that only 316 are not minus 1, Ok.

So we need memory only of that length, 316 times z. Ok what is that z? It gets expanded by z and that many 1s are there in the final matrix, Ok. Or so that is good. So we have done that. We have this R, Ok. Now this R is sort of like a linear entry so the every non minus 1, I have R Ok, so let me show you why it is important.

So the storage for this, Ok storage for this will be R of 1 comma 1 colon 16, right.

(Refer Slide Time: 18:52)

What will be the storage for this? That will be R of 1, 2 comma 1 colon 16 and

(Refer Slide Time: 19:00)

so on, Ok. And this will be R of 3, R of 4 and then there is a minus 1 and if you remember the matrix B, Ok,

(Refer Slide Time: 19:13)

The MATLAB Command Window shows the following code and output:

```

B = NR 1 0 16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~= -1); %number of non -1 in B
>> Slen
Slen =
    316
>> mb * nb
ans =
    3128
>> B(1,1:9)
ans =
    10     5     2    15    -1     4    10    -1    -1
  
```

Below the window, a small video inset shows Prof. Andrew Thangaraj speaking. The slide footer includes the NPTEL logo, the professor's name (PROF. ANDREW THANGARAJ, IIT MADRAS), and the title "Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

there was the minus 1 and then 4, 10, minus 1, minus 1, Ok. Let me go to that once again.

After minus 1, you have 4, 10, minus 1 so on. So this will be R of 2 16, the storage corresponding this will be R of 3 comma 1 colon 16, the storage corresponding to this will be R of 4 comma 1 colon 16. There is no storage here corresponding to minus 1. And the storage corresponding to this is R of 5 comma 1 colon 16, and so on.

This will be R of 6 comma 1 colon 16 and so on, right. And so it will proceed like this. So, Ok so it should be,

(Refer Slide Time: 19:54)

The handwritten diagram illustrates the storage matrix structure for the first row of the base matrix. It shows a sequence of operations and storage requirements:

- Base Matrix:** The first row is $[10, 5, 2, 15, -1, 4, 10, -1, -1]$.
- Storage Matrix:** The storage matrix is a 16×16 identity matrix shifted right by 10 columns.
- Operations:** The diagram shows operations $R[2,1:16]$, $R[3,1:16]$, and $R[4,1:16]$ being applied to the storage matrix.
- Storage Requirements:** The storage requirements are indicated as $R[2,1:16]$, $R[3,1:16]$, and $R[4,1:16]$.

Below the diagram, a small video inset shows Prof. Andrew Thangaraj speaking. The slide footer includes the NPTEL logo, the professor's name (PROF. ANDREW THANGARAJ, IIT MADRAS), and the title "Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

so, so hopefully that is clear to you. So this is how I am going to organize my storage, Ok. So it goes row wise, Ok and wherever there is a minus 1, I will just skip that index. I will go to the next index in this. Is that Ok? Hopefully that is clear, Ok.

So once again remember my storage is just linear. I am storing z values for each non zero non minus 1 entry in the base matrix and, it just goes linear. As you traverse row wise, I will go next memory, next memory, next memory like, Ok, keep that in mind.

So that is number 1. So as I process the first layer like this, I know, I know what to do. So I come here, I have my storage. So I have my access to R. I have the L, the total belief corresponding to this. Remember the L will come out, L is there, right. So that is just for the whole n b into z.

Ok so you will have the first 16, then the next 16 so on, Ok. So you will have the first 16 Ls corresponding to

(Refer Slide Time: 21:01)

The slide contains the following handwritten content:

- NPTEL logo
- 1st row of Base matrix: $[10 \ 5 \ 2 \ 15 \ -1]$
- Annotations: $R[1,1:16]$, $R[2,1:16]$, $R[3,1:16]$, $R[4,1:16]$, $R[5,1:16]$, $R[6,1:16]$, $R[7,1:16]$, $R[8,1:16]$, $R[9,1:16]$, $R[10,1:16]$, $R[11,1:16]$, $R[12,1:16]$, $R[13,1:16]$, $R[14,1:16]$, $R[15,1:16]$, $R[16,1:16]$
- Storage matrix
- 16x16 identity shifted right by 16 columns
- Storage matrix structure: $L = \begin{bmatrix} \text{---} & \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & \text{---} \end{bmatrix}$ with dimensions $mb \times z$
- Inset video: A man in a purple shirt speaking.
- Footer: PROE ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

this, the R will be 1 comma, 1 colon 16. But remember there is this, little bit of shifted thing there, right. So one needs to pay attention to that. When you align it with the row but as far as columns are concerned, it is the same, Ok.

So I am going to subtract. You remember the row processing. I do L minus R first, Ok. That is the subtract operation, layer decoding. And then you do the row processing, Ok. There you

have to align with the rows and then do the row processing. And then the L minus R and the new row get added to get the new total l l r. Ok so that is the operation that we have to do.

So we will do all of that. So one needs to exercise some care when we do this. Nevertheless it is, hopefully it is easy enough to remember, Ok. Alright so that is this picture here.

So hopefully, it is clear to you. You have the total belief L, Ok so you have the total belief here, L which is

(Refer Slide Time: 22:00)

The diagram illustrates the construction of a 'Total belief' matrix L. It starts with a 'Base matrix' (1st row: 10, 15, 2, 15, -1) and a 'Storage matrix' (16x16 identity shifted right by 10 columns). The diagram shows the addition of the base matrix to the storage matrix, resulting in a matrix with two 16x16 blocks. The final 'Total belief' matrix L is shown as a block matrix with dimensions m_b x 2.

in one vector and then you have this R matrix which is storing sequentially as you go along the rows, what your R values, initially it is all zero but still you should store that and then you do, then you have to do row processing, Ok.

Row processing, row processing involves L minus R. That is the first step.

(Refer Slide Time: 22:24)

NPTEL

Storage matrix

16×16 identity shifted right by 10 columns

4 10 -1...
R(5,11) R(6,12)

Total delay $\left[\begin{array}{c|c|c} & & \\ \hline & 16 & \\ \hline & & \dots \\ \hline & & 16 \times 2 \end{array} \right]$

Row processing:
1) $L - R$

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Next minsum, row aligned, Ok so you have to do a row alignment here.

(Refer Slide Time: 22:32)

NPTEL

Storage matrix

16×16 identity shifted right by 10 columns

4 10 -1...
R(5,11) R(6,12)

Total delay $\left[\begin{array}{c|c|c} & & \\ \hline & 16 & \\ \hline & & \dots \\ \hline & & 16 \times 2 \end{array} \right]$

Row processing:
1) $L - R$
2) min sum (row aligned)

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Remember, why do I have to do a row alignment? Hopefully it is clear to you because,

(Refer Slide Time: 22:35)

so while R starts like this; that is not aligned in the same row, Ok. So the first thing is what is aligned.

So you have to shift by 10 and then only you will get alignment in the row. Ok, those are the corresponding things that you have to work with. Ok and final thing is L minus R plus the, minsum value, Ok minsum updated value, Ok.

(Refer Slide Time: 23:00)

This is what I have to implement. And there is this row alignment which I have to be careful about, Ok. So to actually do the row processing, I like to sometimes pull in the values from the storage into some temporary storage that I will have and then work with that. Ok, so there are various ways to do that and one, so, so this also makes sense.

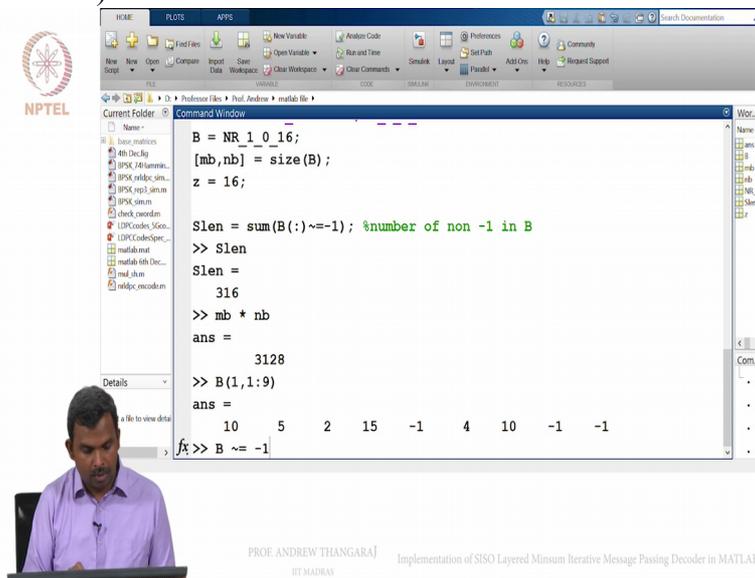
Because usually the storage is in some, when you actually implement in, let us say hardware, the storage is in some memory and you want to pull it in some registers, into a small operating unit and do the calculation and then write back into memory.

So this is something that you want to do to make sure things work cleanly. So I am going to use some registers, temporary storage for pulling in these, these R values. L is Ok; L is just available to me all the time. R values I am going to pull from memory into my sort of some temporary storage, do the processing and then store it back, Ok. So that will be my flow, Ok.

So let us see how to get this done. Ok, not getting the right thing, Ok there you go. So I need to define that also, Ok. So how many do I need? So for that I need to know the maximum number of 1s in a row, Ok. In the expanded base matrix, number of, maximum number of non minus 1s in the rows of B, Ok. So for that one can write small piece of code.

So if you look at B not equal to minus 1,

(Refer Slide Time: 24:40)



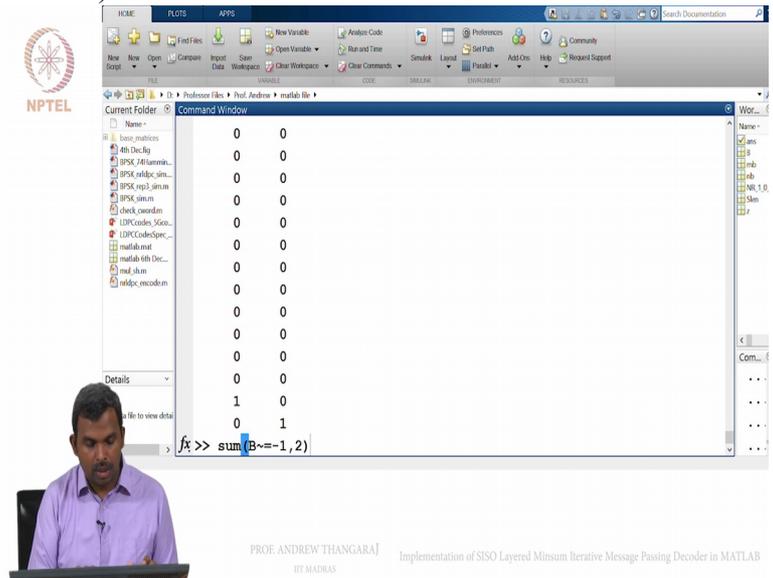
The screenshot shows the MATLAB environment with the following code in the script editor:

```
B = NR_1_0_16;  
[mb,nb] = size(B);  
z = 16;  
  
Slen = sum(B(:)~= -1); %number of non -1 in B  
>> Slen  
Slen =  
316  
>> mb * nb  
ans =  
3128  
>> B(1,1:9)  
ans =  
10 5 2 15 -1 4 10 -1 -1  
fx>> B ~= -1
```

The command window shows the execution of these commands and the resulting values. Below the screenshot, there is a small video inset of a man in a purple shirt and a slide footer with the text: "PROF. ANDREW THANGARAJ IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

this will give me a matrix, Ok it is tough to see that. So if you add up B not equal to minus 1 comma 2

(Refer Slide Time: 24:50)



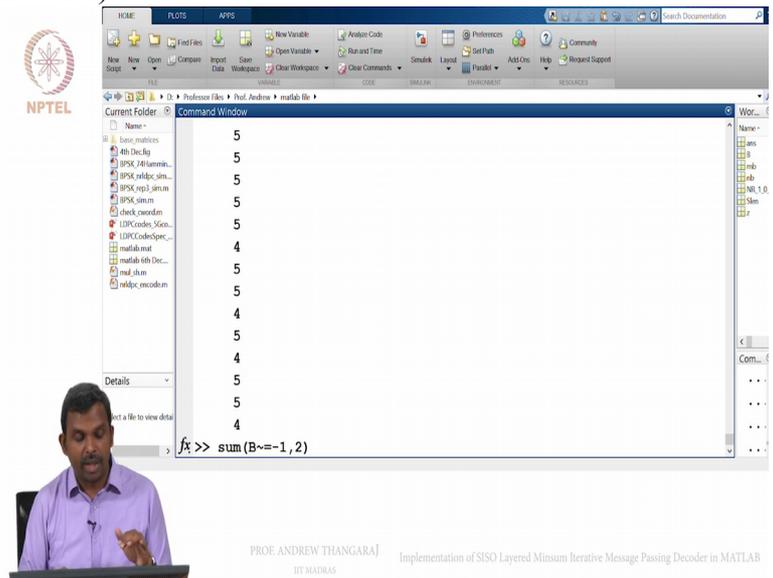
The image shows a MATLAB Command Window with a matrix of zeros and a sum command. The matrix is a 10x2 grid of zeros. The command window shows the following text:

```
0 0  
0 0  
0 0  
0 0  
0 0  
0 0  
0 0  
0 0  
0 0  
0 0  
1 0  
0 1  
fx>> sum(B==-1,2)
```

Below the MATLAB window, there is a small video inset of Prof. Andrew Thangaraj and a slide footer that reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok

(Refer Slide Time: 24:50)



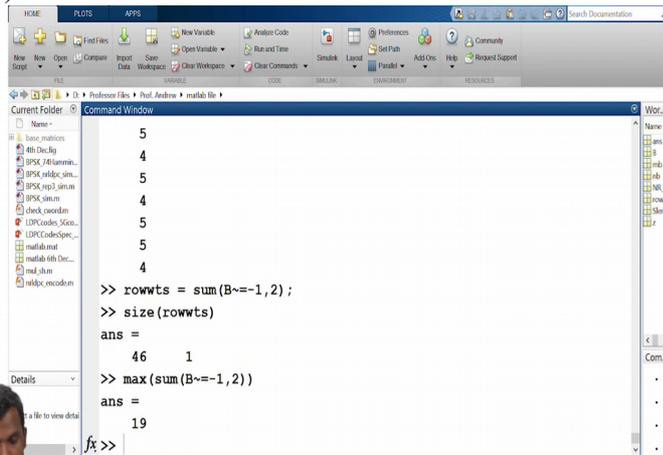
The image shows a MATLAB Command Window with a column vector of values and a sum command. The vector contains the values 5, 5, 5, 5, 5, 5, 4, 5, 5, 4, 4, 5, 5, 4. The command window shows the following text:

```
5  
5  
5  
5  
5  
5  
4  
5  
5  
4  
4  
5  
5  
4  
fx>> sum(B==-1,2)
```

Below the MATLAB window, there is a small video inset of Prof. Andrew Thangaraj and a slide footer that reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

so this will give me, this will give me the row weights. Ok so if you want, I can define this as row weights.

(Refer Slide Time: 25:23)



The image shows a MATLAB Command Window with the following text:

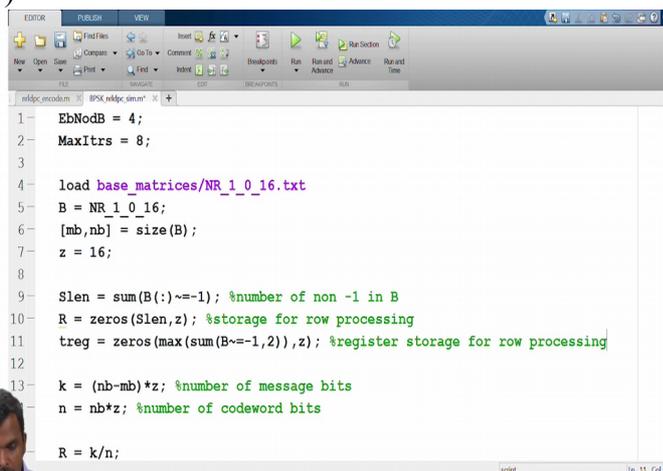
```
5
4
5
4
5
5
4
>> rowwts = sum(B~= -1, 2);
>> size(rowwts)
ans =
    46     1
>> max(sum(B~= -1, 2))
ans =
    19
```

Below the window, a small video inset shows Prof. Andrew Thangaraj. The slide footer reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok. So there are 19, that is the maximum weight that you have in this matrix, Ok. So that is the number there. So one can do that also.

So, so, so, so let me define the temporary storage. I will call it t reg, temporary registers, Ok, zeroes of max of B not equal to sum of B not equal to minus 1 comma 2. I do not think I need this number anywhere else so that is why I am not worrying too much about it. That is it, Ok. So this is the, the storage, register storage I will call it for row processing.

(Refer Slide Time: 26:19)



The image shows a MATLAB Editor window with the following code:

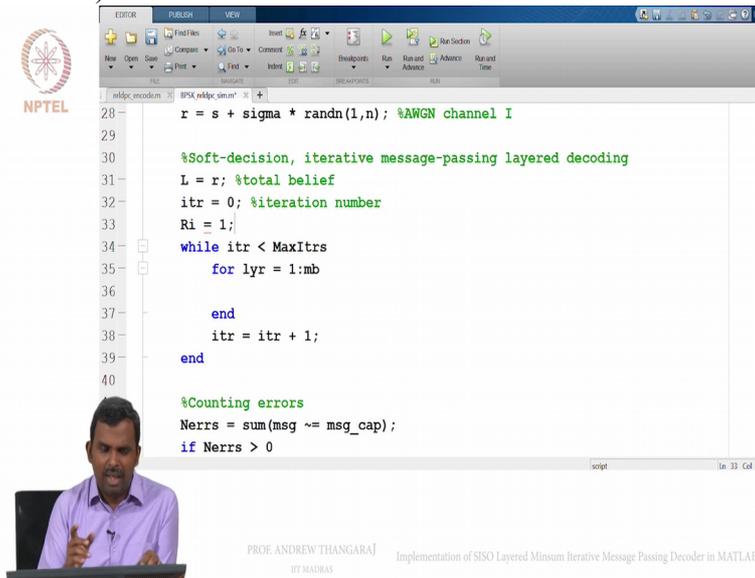
```
1- EbNodB = 4;
2- MaxIters = 8;
3
4- load base_matrices/NR_1_0_16.txt
5- B = NR_1_0_16;
6- [mb,nb] = size(B);
7- z = 16;
8
9- Slen = sum(B(:,)~= -1); %number of non -1 in B
10- R = zeros(Slen,z); %storage for row processing
11- treg = zeros(max(sum(B~= -1, 2)), z); %register storage for row processing
12
13- k = (nb-mb)*z; %number of message bits
    n = nb*z; %number of codeword bits
    R = k/n;
```

Below the editor, a small video inset shows Prof. Andrew Thangaraj. The slide footer reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

So all the row processing I will use t reg for, Ok. So let us do that.

Ok so now I come to the layer. So I have to now read the, the storage, the R storage into my t reg, Ok. So I also, it also depends on what layer I am in, Ok. So there are various ways to, to do this. So inside a layer, I am inside each layer, so I will have to keep a count on where I am in R. So I will, I will keep that also as R count equals, I will call it as simpler notation R i equals, we will have

(Refer Slide Time: 27:10)



The screenshot shows a MATLAB script in an editor window. The code includes comments for AWGN channel simulation and iterative decoding. A small video inset shows Prof. Andrew Thangaraj speaking.

```

28 r = s + sigma * randn(1,n); %AWGN channel I
29
30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36
37     end
38     itr = itr + 1;
39 end
40
41 %Counting errors
42 Nerrs = sum(msg ~= msg_cap);
43 if Nerrs > 0

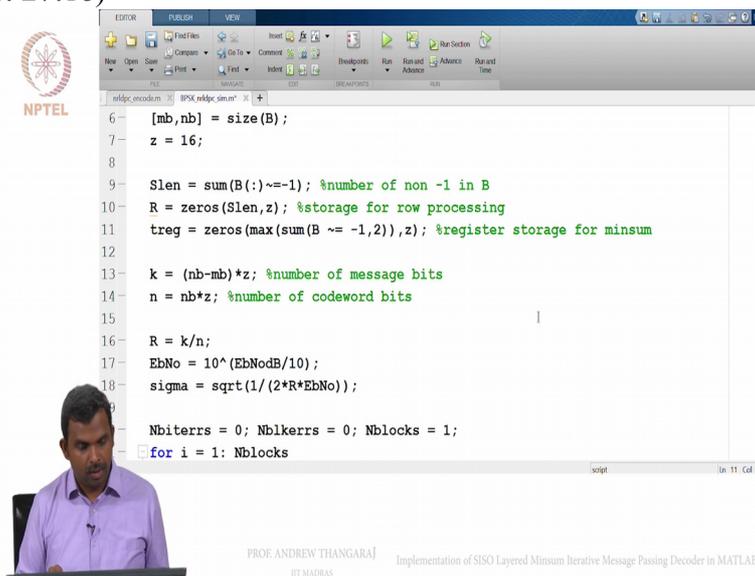
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

this one, does not matter.

So this is register storage for minsum.

(Refer Slide Time: 27:18)



The screenshot shows a MATLAB script in an editor window. The code initializes variables for register storage and message bits. A small video inset shows Prof. Andrew Thangaraj speaking.

```

6 [mb,nb] = size(B);
7 z = 16;
8
9 Slen = sum(B(:)~= -1); %number of non -1 in B
10 R = zeros(Slen,z); %storage for row processing
11 treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum
12
13 k = (nb-mb)*z; %number of message bits
14 n = nb*z; %number of codeword bits
15
16 R = k/n;
17 EbNo = 10^(EbNodB/10);
18 sigma = sqrt(1/(2*R*EbNo));
19
20 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
21 for i = 1: Nblocks

```

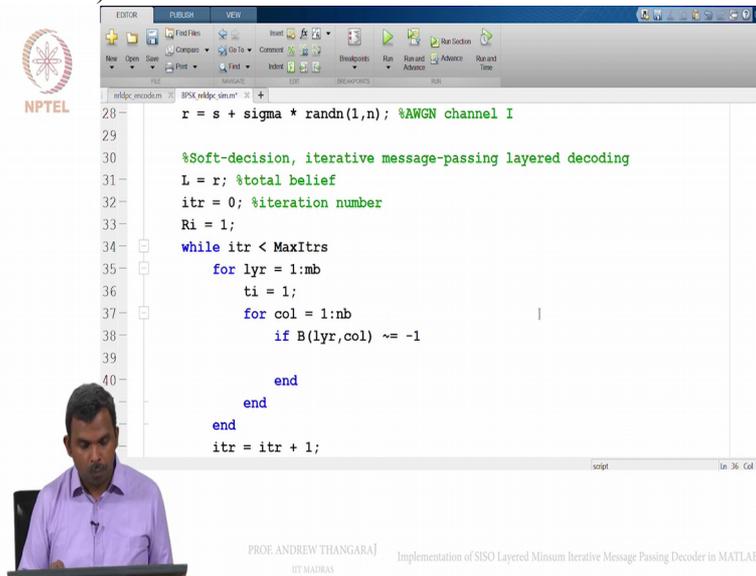
PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

That is good to have. So let us go through. So I am going to go through here for each layer and for each column in the base matrix. I am checking if it is minus 1 or not. If it is not minus

1, I can store it into my register. But I will store it after doing subtraction and, and taking care of the index also.

So let me do this bit more carefully. So I will call it t_i equals 1.

(Refer Slide Time: 27:47)



The screenshot shows a MATLAB script editor with the following code:

```

28 r = s + sigma * randn(1,n); %AWGN channel I
29
30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 1;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39
40             end
41         end
42     end
43     itr = itr + 1;

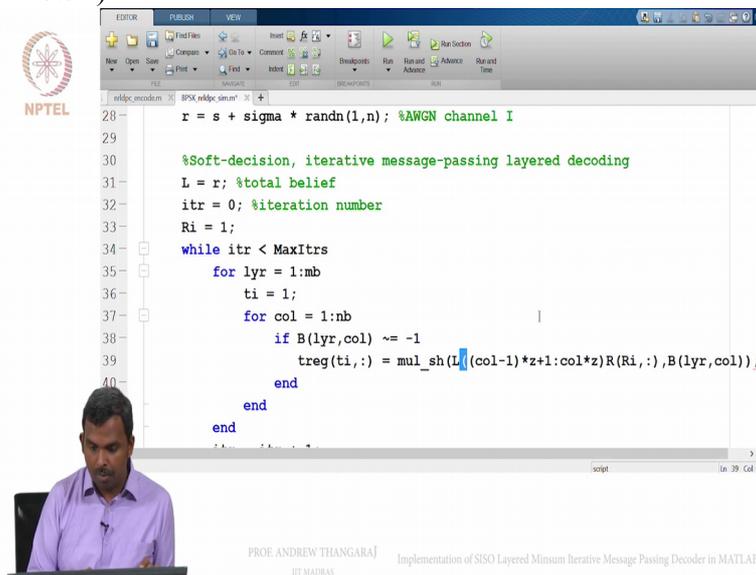
```

Below the code, a small video inset shows Prof. Andrew Thangaraj speaking. The slide footer identifies him as Prof. Andrew Thangaraj, IIT Madras, and the title as 'Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB'.

And then if it is not minus 1, my t_i reg of t_i comma colon equals, I need to do a mul shift of R_i comma colon comma B of layer comma col, Ok.

So I am shifting it and store. What should I shift? Not R , I need to do L of, the column is 1 so it needs to be col minus 1 multiplied by z plus 1 colon col into z

(Refer Slide Time: 28:32)



The screenshot shows the same MATLAB script editor as before, but with an additional line of code in the inner loop:

```

38             if B(lyr,col) ~= -1
39                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)R(Ri,:), B(lyr,col));
40             end
41         end
42     end
43     itr = itr + 1;

```

Below the code, a small video inset shows Prof. Andrew Thangaraj speaking. The slide footer identifies him as Prof. Andrew Thangaraj, IIT Madras, and the title as 'Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB'.

minus R_i and then comma this, Ok. So let me just go through this. I just typed out a lot of things.

So you can see what I am doing here. I am taking the corresponding L . From L if I might,

(Refer Slide Time: 28:45)

NPTEL

```

28 r = s + sigma * randn(1,n); %AWGN channel I
29
30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 1;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 trellis(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:), B(lyr,col));
40             end
41         end
42     end

```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

a column c o l , then L of column minus 1 into z plus 1 colon column into z is the corresponding block of total belief from the entire L vector. And what is my R ? That is R of R_i

(Refer Slide Time: 29:01)

NPTEL

```

28 r = s + sigma * randn(1,n); %AWGN channel I
29
30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 1;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 trellis(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:), B(lyr,col));
40             end
41         end
42     end

```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

comma 1.

So R_i is my index in the, in the message, the storage register for the row processing and there I take the R_i th one but after I subtract I need to shift it. The shifting is to row align, Ok so that is the mul shift that we had. Even in encoder we saw this operation. I am going to shift it by the entry in the B, Ok. So that is one step lets me do this.

But then after I do this, I have to increment my t_i

(Refer Slide Time: 29:33)

The screenshot shows a MATLAB editor window with the following code:

```

28 r = s + sigma * randn(1,n); %AWGN channel I
29
30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 1;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
40                 ti = ti + 1;
41             end
42         end
43     end
44 end
  
```

Below the code, a small video inset shows Prof. Andrew Thangaraj speaking. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

and increment my R_i .

(Refer Slide Time: 29:37)

The screenshot shows the same MATLAB editor window as the previous slide, but with an additional line of code added inside the innermost for loop:

```

38             if B(lyr,col) ~= -1
39                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
40                 ti = ti + 1;
41                 Ri = Ri + 1;
42             end
43         end
44     end
45 end
  
```

Below the code, a small video inset shows Prof. Andrew Thangaraj speaking. The text below the video reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Is it Ok?

So once I did this, I have got all the values from the storage for the row processing into my register for minsum, Ok. Hopefully I will go through this once again. For every layer, I am look at every block column, I am seeing if it is minus 1 or not. If it is not minus 1, I have to process it. What do I do? This step actually does subtraction and row alignment.

(Refer Slide Time: 30:09)

The screenshot shows a MATLAB editor window with the following code:

```

30 %Soft-decision, iterative message-passing layered decoding
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 1;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 1;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 %Subtraction and row alignment
40                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col));
41                 ti = ti + 1;
42                 Ri = Ri + 1;
43             end
44         end
45     end
46 end

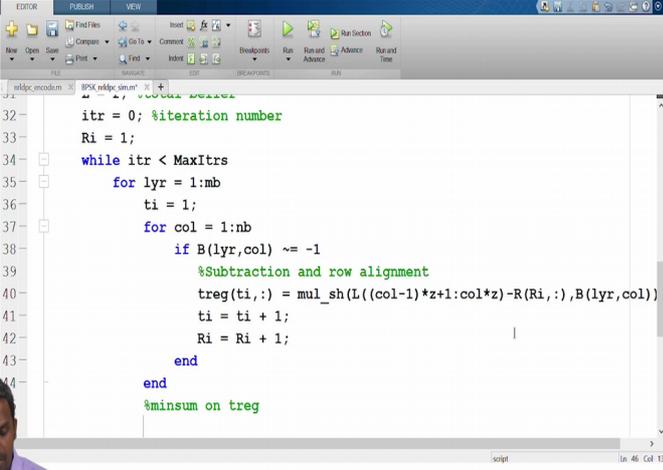
```

Below the code, a small video inset shows a man in a purple shirt speaking. The NPTEL logo is in the top left. At the bottom, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok, so it brings in the corresponding value from the row processing, row storage R and the corresponding value from the total belief. It subtracts the, subtracts the two and then shifts it by the actual value in the shift, in the, in the base matrix and then stores in t reg.

So at the end of this loop, once I am done with the every column here, I have all the values read, brought into, brought into this t reg register and they are all nicely row aligned, Ok. So now I am ready to do minsum, Ok, minsum on t reg, Ok.

(Refer Slide Time: 30:48)



```
itr = 0; %iteration number
Ri = 1;
while itr < MaxItrs
    for lyr = 1:mb
        ti = 1;
        for col = 1:nb
            if B(lyr,col) ~= -1
                %Subtraction and row alignment
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
                ti = ti + 1;
                Ri = Ri + 1;
            end
        end
        %minsum on treg
    end
end
```

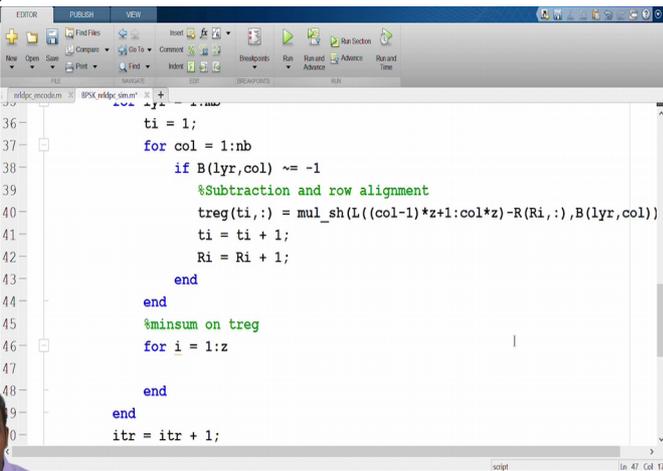


PROE ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

On every entry in t reg I have to do minsum and that is, I will just call it for i equals 1 colon z, Ok

(Refer Slide Time: 30:58)



```
ti = 1;
for col = 1:nb
    if B(lyr,col) ~= -1
        %Subtraction and row alignment
        treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
        ti = ti + 1;
        Ri = Ri + 1;
    end
end
%minsum on treg
for i = 1:z
end
itr = itr + 1;
```



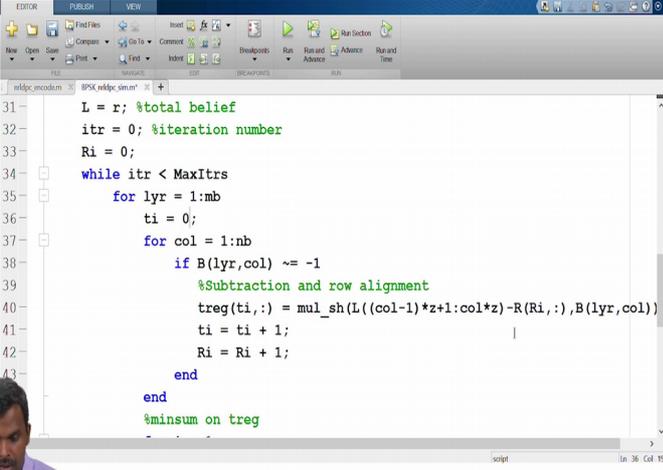
PROE ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

I have to do minsum, Ok. So, so doing minsum, you can do in various ways. I am doing a simple loop here. You can do the without loops also if you like. I am going to do it just for simplicity; I am going to do it with loops.

And, just for, Ok so maybe this is, this is slightly, maybe we will keep it as 0 here just for,

(Refer Slide Time: 31:30)



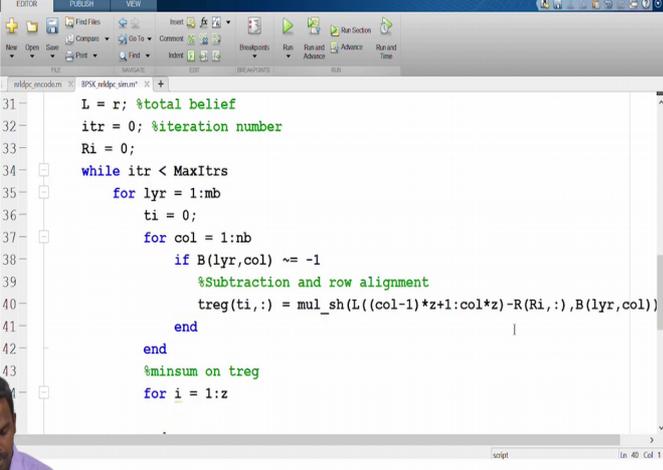
```
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 0;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 0;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 %Subtraction and row alignment
40                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
41                 ti = ti + 1;
42                 Ri = Ri + 1;
43             end
44         end
45     end
46     %minsum on treg
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

I will do this

(Refer Slide Time: 31:37)



```
31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 0;
34 while itr < MaxItrs
35     for lyr = 1:mb
36         ti = 0;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 %Subtraction and row alignment
40                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
41             end
42         end
43     end
44     %minsum on treg
45     for i = 1:z
```

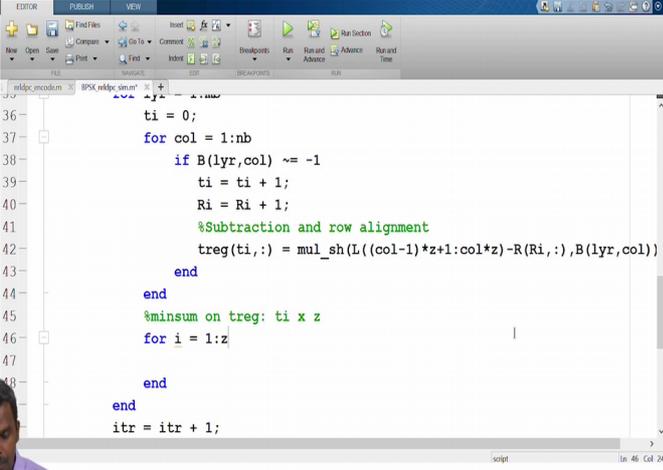
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

in a slightly reverse way, Ok. So this is better because if I start at 0 and add 1 before doing this, then I know t_i is the total number of non-zero entries at that point. So I do not have to worry about that. So t_i will be my total number of non-zero entries, Ok.

So t_i , this is $t_i \times z$, Ok.

(Refer Slide Time: 31:59)



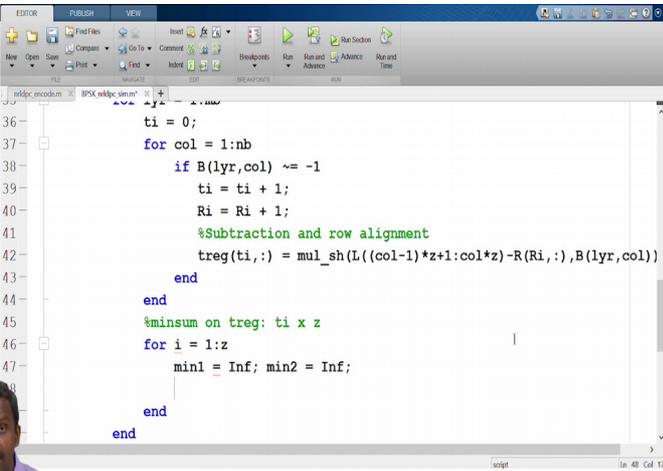
```
36 ti = 0;
37 for col = 1:nb
38     if B(lyr,col) ~= -1
39         ti = ti + 1;
40         Ri = Ri + 1;
41         %Subtraction and row alignment
42         treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43     end
44 end
45 %minsum on treg: ti x z
46 for i = 1:z
47     |
48 end
end
itr = itr + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So there are t_i values and for every value in every, and that is all aligned nicely, so I just have to do the, do the row processing, Ok. So how do I, how I go about doing this? So so that is clear, so I need my min 1 and min 2, so I will set min 1 to be infinity, min 2 to be infinity. MATLAB allows me to do that,

(Refer Slide Time: 32:27)



```
36 ti = 0;
37 for col = 1:nb
38     if B(lyr,col) ~= -1
39         ti = ti + 1;
40         Ri = Ri + 1;
41         %Subtraction and row alignment
42         treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43     end
44 end
45 %minsum on treg: ti x z
46 for i = 1:z
47     min1 = Inf; min2 = Inf;
48     |
49 end
end
```

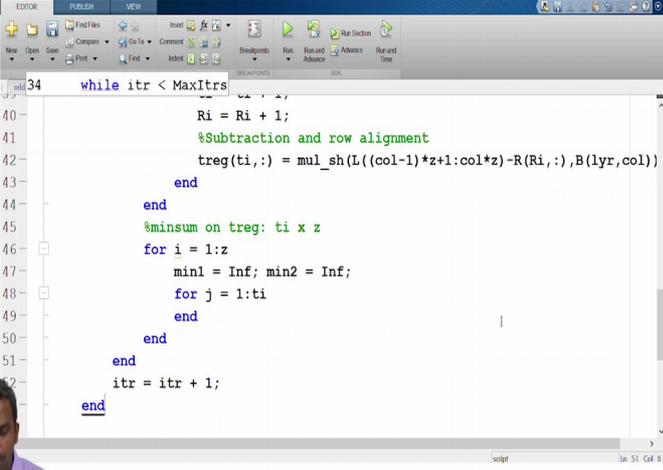
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

so large number Ok.

And then I will just process, Ok. So how do I process for j equals 1 colon t_i ? I will end it.

(Refer Slide Time: 32:38)

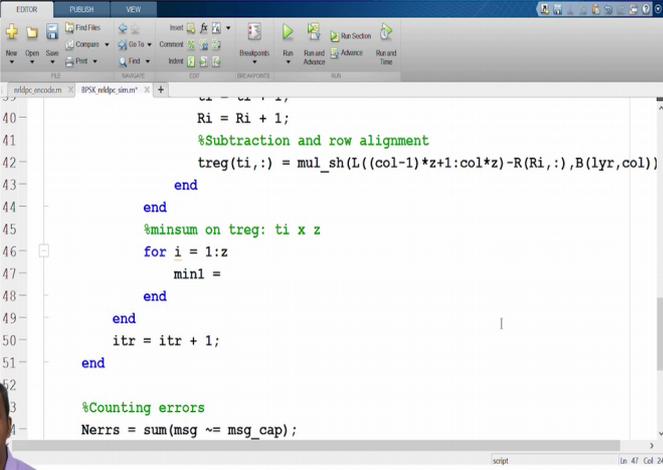


```
34 while itr < MaxItrs
35     Ri = Ri + 1;
36     %Subtraction and row alignment
37     treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
38
39     end
40
41     end
42
43     %minsum on treg: ti x z
44     for i = 1:z
45         min1 = Inf; min2 = Inf;
46         for j = 1:ti
47             end
48         end
49     end
50
51     end
52     itr = itr + 1;
53 end
```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So loops actually are quite slow and bad in MATLAB so you should try to not use loops but I am just using loops for simplicity. So maybe we can avoid this loop here. So let us, write it that,

(Refer Slide Time: 32:53)



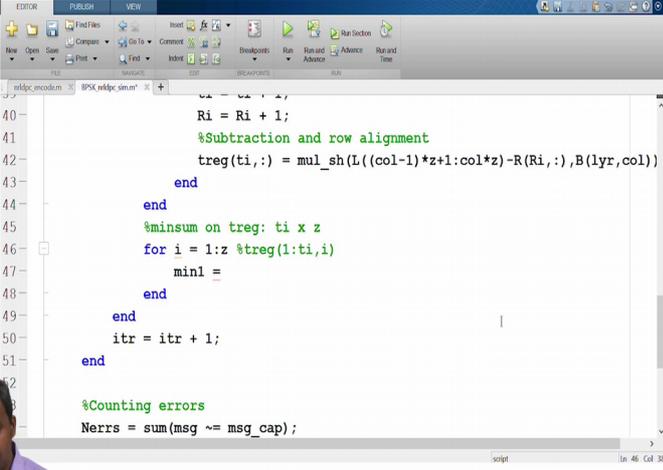
```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43
44 end
45
46 %minsum on treg: ti x z
47 for i = 1:z
48     min1 =
49     end
50 end
51 itr = itr + 1;
52 end
53
54 %Counting errors
55 Nerrs = sum(msg ~= msg_cap);
```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok.

So now I need to find, so remember I am processing the ith, so t reg of colon comma 1 colon t i comma i, Ok so the ith column

(Refer Slide Time: 33:08)



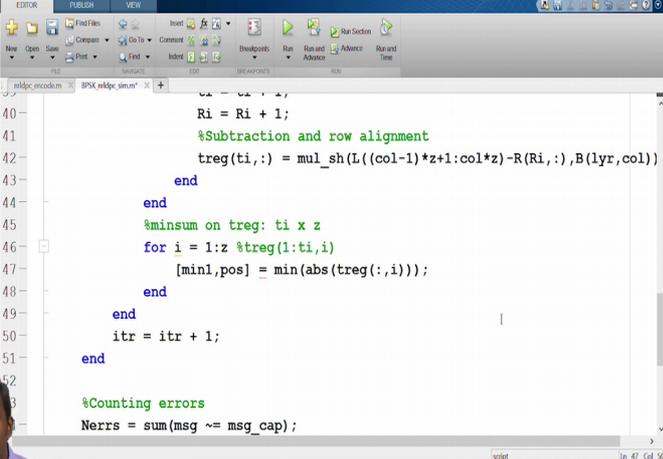
```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i = 1:z %treg(1:ti,i)
47 minl =
48 end
49 end
50 itr = itr + 1;
51 end
52 %Counting errors
53 Nerrs = sum(msg ~= msg_cap);
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

in t reg is what I am processing, Ok. So I have to find the minimum value among, absolute minimum value in the ith column of t i and then the second absolute minimum value then the product of the signs and then I flip based on that, right.

So that is what. I need to find the min. And not only the min, I also need the position, right. pos equals min of a b s of t i, t reg of colon comma i,

(Refer Slide Time: 33:43)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i = 1:z %treg(1:ti,i)
47 [minl,pos] = min(abs(treg(:,i)));
48 end
49 end
50 itr = itr + 1;
51 end
52 %Counting errors
53 Nerrs = sum(msg ~= msg_cap);
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok.

So this gives me the first minimum. Is that Ok? So the first minimum, otherwise it is complaining over i, oh, Ok sorry it is i 1. So looks like I have used another index for i. There is an i for the ith block. So alright. So i 1 is something, Ok so this is the first minimum,

(Refer Slide Time: 34:13)

The screenshot shows a MATLAB script with the following code:

```

40     Ri = Ri + 1;
41     %Subtraction and row alignment
42     treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43     end
44     end
45     %minsum on treg: ti x z
46     for il = 1:z %treg(1:ti,i)
47         [minl,pos] = min(abs(treg(:,il))); %first minimum
48     end
49     end
50     itr = itr + 1;
51     end
52
53     %Counting errors
54     Nerrs = sum(msg ~= msg_cap);

```

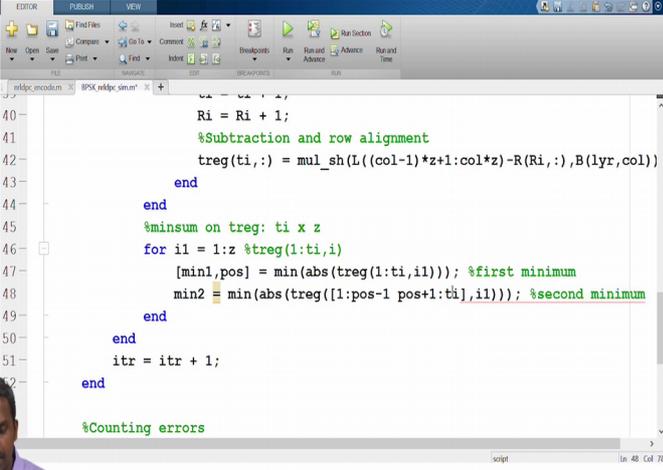
At the bottom of the slide, it reads: PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB.

absolute minimum of course.

So I am ready to find the second minimum. For the second minimum, one should not use the value in pos. So min of abs of t reg of, I also not supposed to use the first value, so 1 colon pos minus 1 and then pos plus 1 colon end, not end, colon t i so it needs to be careful here, let us not use 1 colon t i, it is very important; comma i 1.

Ok so this should give me the second minimum. We will check it out if I made a mistake,

(Refer Slide Time: 34:52)



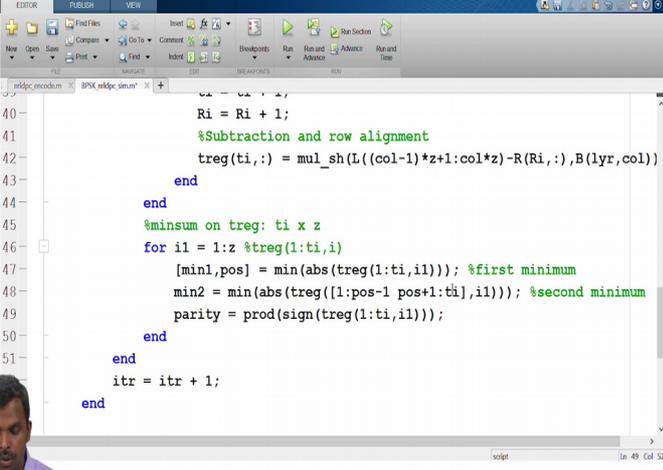
```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i)
47 [min1,pos] = min(abs(treg(1:ti,i))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
49 end
50 end
51 itr = itr + 1;
52 end
%Counting errors
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

we will correct it. So this should give second minimum, Ok.

So the first minimum and second minimum are set, it is very nice, but then what about the sign? The sign, the sign is I need this parity. I take product of sign of t reg of 1 comma t i comma i 1, Ok. So that is parity for you and then

(Refer Slide Time: 35:26)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i)
47 [min1,pos] = min(abs(treg(1:ti,i))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
49 parity = prod(sign(treg(1:ti,i1)));
50 end
51 end
52 itr = itr + 1;
53 end
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

one can start, so it is good to keep the signs also.

So maybe what I will do is I will store the signs separately. I will call it S.

(Refer Slide Time: 35:55)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i)
47 [min1,pos] = min(abs(treg(1:ti,i))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i))); %second minimum
49 S = sign(treg(1:ti,i))
50 parity = prod();
51 end
52 end
53 itr = itr + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

These are the signs. Product of S is the parity,

(Refer Slide Time: 35:58)

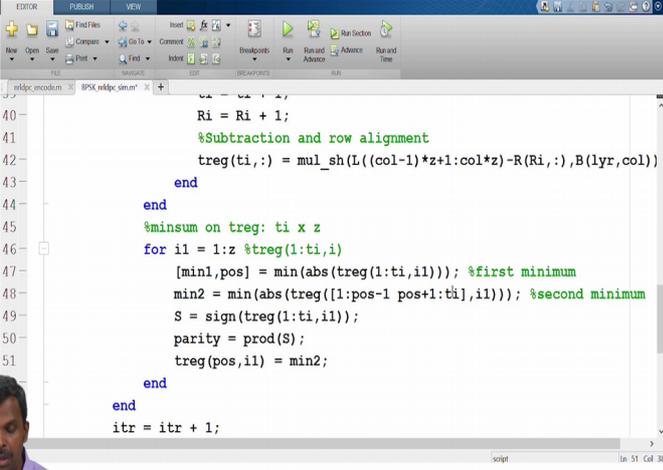


```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i)
47 [min1,pos] = min(abs(treg(1:ti,i))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i))); %second minimum
49 S = sign(treg(1:ti,i));
50 parity = prod(S);
51 end
52 end
53 itr = itr + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok and then t reg of pos comma i 1 equals min 2, Ok some setting

(Refer Slide Time: 36:10)

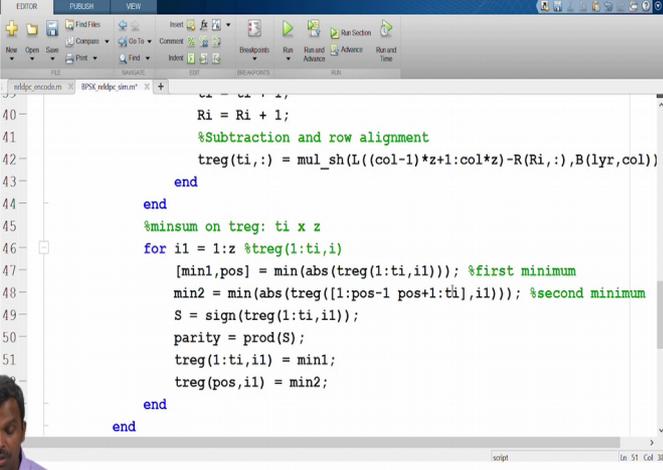
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

wherever the minimum was, I am setting the second minimum, t reg of, so actually it is better to do this in the, in the wrong, in the right way.

First do min t reg of 1 colon t i comma i 1 equals min 1. Let us do the absolute minimum

(Refer Slide Time: 36:28)

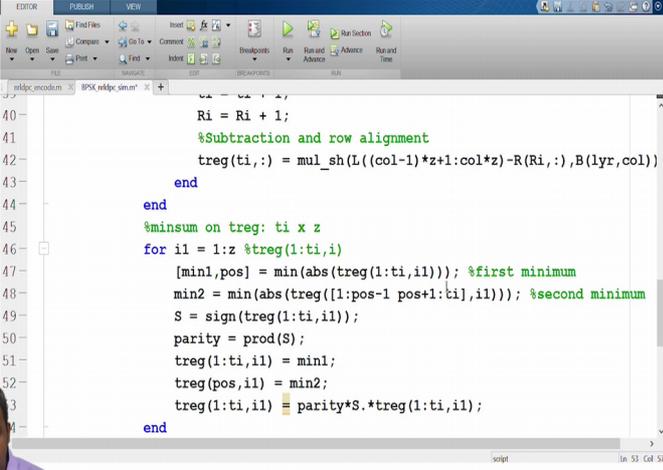
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

and then position alone you replace with min 2. So this is slightly easy way to do it. And what about the signs? t reg of 1 colon t i comma i 1 equals s prod parity times s dot into t reg of 1 colon t i comma i.

So first we assign the absolute

(Refer Slide Time: 36:53)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for il = 1:z %treg(1:ti,il)
47 [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
49 S = sign(treg(1:ti,il));
50 parity = prod(S);
51 treg(1:ti,il) = min1;
52 treg(pos,il) = min2;
53 treg(1:ti,il) = parity*S.*treg(1:ti,il);
54 end
```

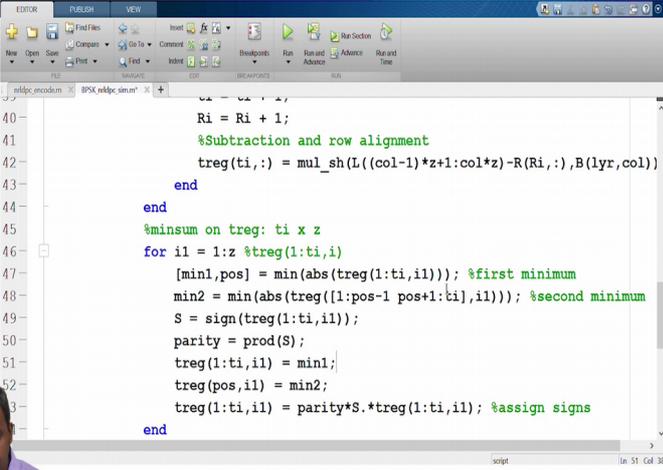


PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

values, Ok so assign min 1 to everything and then the minimum, first minimum position gets replaced by min 2. So at this point only absolute values are there. And then we assign the signs, Ok.

(Refer Slide Time: 37:07)



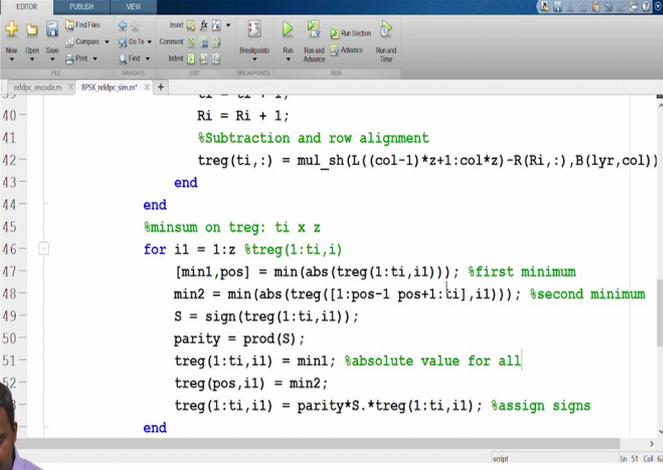
```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for il = 1:z %treg(1:ti,il)
47 [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
49 S = sign(treg(1:ti,il));
50 parity = prod(S);
51 treg(1:ti,il) = min1;
52 treg(pos,il) = min2;
53 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
54 end
```



PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

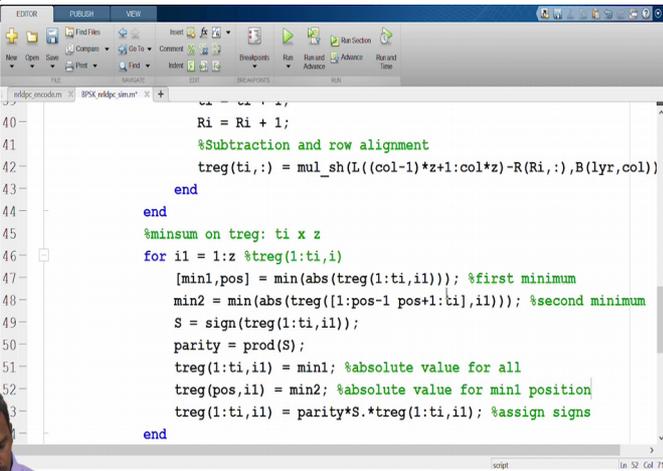
(Refer Slide Time: 37:15)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i1)
47 [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
49 S = sign(treg(1:ti,i1));
50 parity = prod(S);
51 treg(1:ti,i1) = min1; %absolute value for all
52 treg(pos,i1) = min2;
53 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
54 end
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

(Refer Slide Time: 37:22)



```
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col))
43 end
44 end
45 %minsum on treg: ti x z
46 for i1 = 1:z %treg(1:ti,i1)
47 [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
49 S = sign(treg(1:ti,i1));
50 parity = prod(S);
51 treg(1:ti,i1) = min1; %absolute value for all
52 treg(pos,i1) = min2; %absolute value for min1 position
53 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
54 end
```

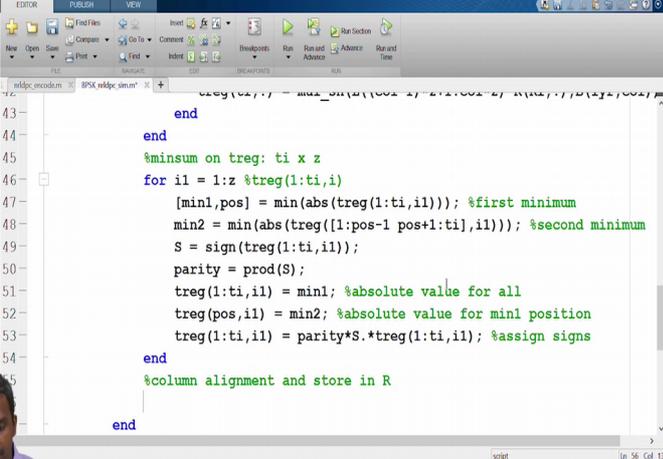
PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok, so hopefully this is clear enough. Remember once again we did subtraction and row alignment and stored the values in t reg. And now we just have to process the values in t reg for minsum operation. How do we do this?

We take the t i values and do minsum on each of those values, Ok. They are already aligned now so we do not have to bother aligning here. We just do, for each, each of the expanded position, expanded rows we do minsum, Ok.

After you have done with minsum, you have to write back, Ok. So column alignment and store in R,

(Refer Slide Time: 38:14)



PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

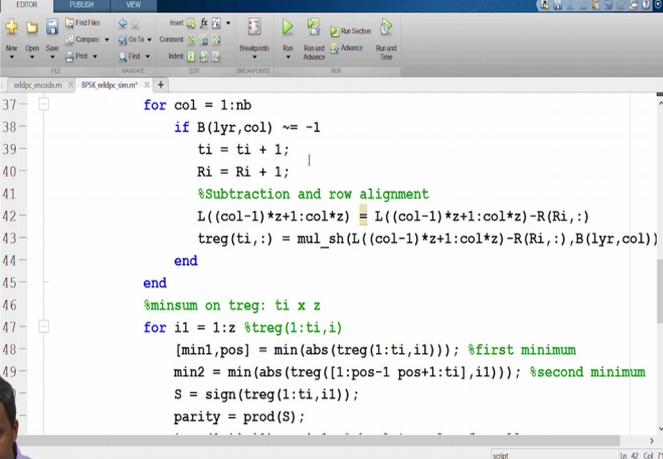
```
43 end
44 end
45 %minsum on treg: ti x z
46 for il = 1:z %treg(1:ti,i)
47 [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
48 min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
49 S = sign(treg(1:ti,il));
50 parity = prod(S);
51 treg(1:ti,il) = min1; %absolute value for all
52 treg(pos,il) = min2; %absolute value for min1 position
53 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
54 end
55 %column alignment and store in R
end
```

Ok. So for that there is an addition also, column alignment, addition and store in R. So one needs to be a little bit careful here. Remember, see when we did the subtraction, the subtracted values, Ok

So we did row processing on the subtracted values but then after the row processing we should not be adding into L. We should add to this. So, so one of the things to do here is, what are the things to do here which I did not do is the assignment of the subtraction.

So I think it is good to do this. Let me. So this is important.

(Refer Slide Time: 39:09)



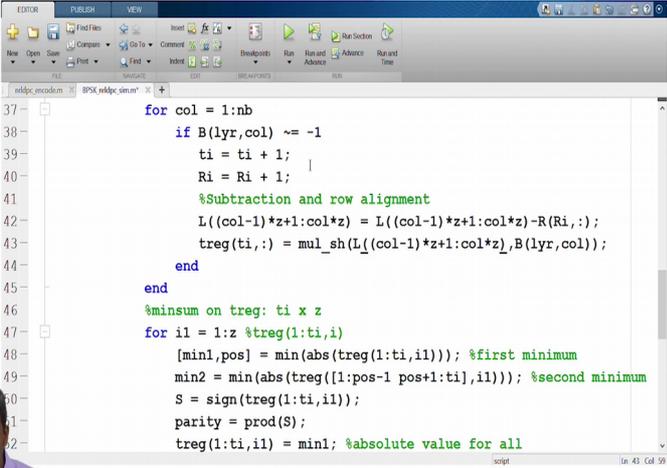
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

```
37 for col = 1:nb
38 if B(lyr,col) ~= -1
39 ti = ti + 1;
40 Ri = Ri + 1;
41 %Subtraction and row alignment
42 L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z)-R(Ri,:),B(lyr,col));
44 end
45 end
46 %minsum on treg: ti x z
47 for il = 1:z %treg(1:ti,i)
48 [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
49 min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
50 S = sign(treg(1:ti,il));
51 parity = prod(S);
```

We need to do this. So once you subtract, you do not have to subtract again, Ok.

(Refer Slide Time: 39:15)

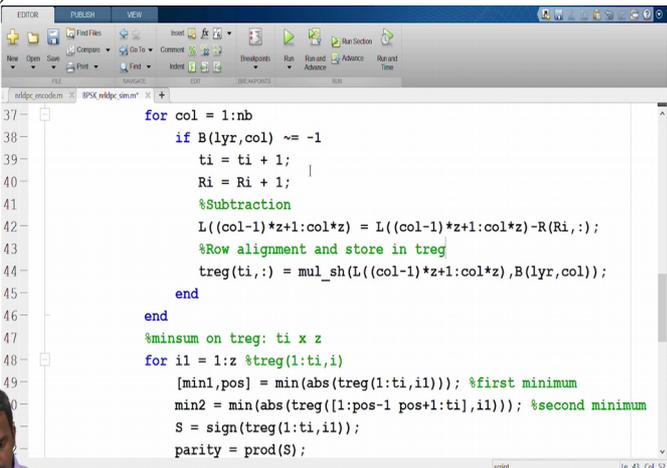


```
37 for col = 1:nb
38     if B(lyr,col) ~= -1
39         ti = ti + 1;
40         Ri = Ri + 1;
41         %Subtraction and row alignment
42         L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43         treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
44     end
45 end
46 %minsum on treg: ti x z
47 for il = 1:z %treg(1:ti,il)
48     [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
49     min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
50     S = sign(treg(1:ti,il));
51     parity = prod(S);
52     treg(1:ti,il) = min1; %absolute value for all
```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So this is nice, Ok. So we did subtraction and then row alignment goes here.

(Refer Slide Time: 39:32)



```
37 for col = 1:nb
38     if B(lyr,col) ~= -1
39         ti = ti + 1;
40         Ri = Ri + 1;
41         %Subtraction
42         L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43         %Row alignment and store in treg
44         treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
45     end
46 end
47 %minsum on treg: ti x z
48 for il = 1:z %treg(1:ti,il)
49     [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
50     min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum
51     S = sign(treg(1:ti,il));
52     parity = prod(S);
```

PROF ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

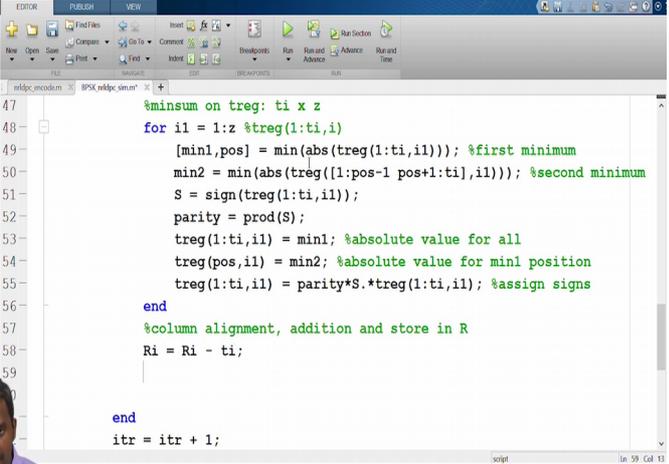
Ok so this is pretty clean. So our total belief has now been subtracted with R and we stored it in, we did the storage in, in the t reg. And then we did minsum. And once we did minsum, we can now add, Ok, column align, add and store back in R.

So I think this is the layer decoding algorithm. Hopefully this part is clear. So it is doing, it is not doing the subtraction and storing carefully. So I think this should be Ok, alright.

So now we do the column alignment and add and this needs to go on for every column, right. So this is, this will go like, Ok. So I think, this is, this is fine. Ok. Ok. So, so we will do the index again.

So R_i is going to be $R_i - t_i$.

(Refer Slide Time: 40:45)

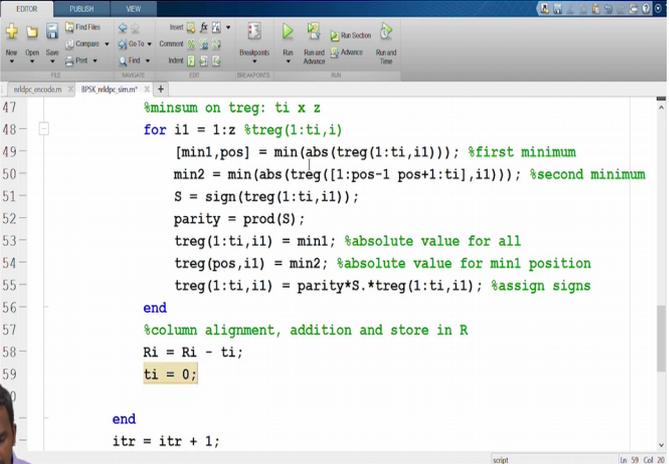
```

47 %minsum on treg: ti x z
48 for i1 = 1:z %treg(1:ti,i)
49 [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
50 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
51 S = sign(treg(1:ti,i1));
52 parity = prod(S);
53 treg(1:ti,i1) = min1; %absolute value for all
54 treg(pos,i1) = min2; %absolute value for min1 position
55 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59
60 end
61 itr = itr + 1;
    
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok snap back at the first index. t_i , we need t_i

(Refer Slide Time: 40:51)

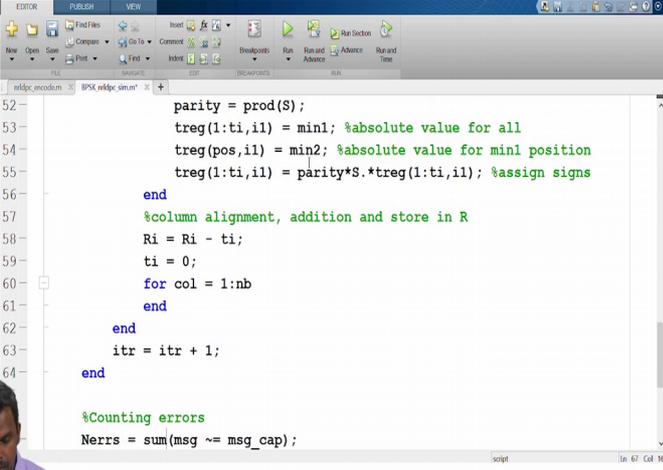
```

47 %minsum on treg: ti x z
48 for i1 = 1:z %treg(1:ti,i)
49 [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
50 min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
51 S = sign(treg(1:ti,i1));
52 parity = prod(S);
53 treg(1:ti,i1) = min1; %absolute value for all
54 treg(pos,i1) = min2; %absolute value for min1 position
55 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 end
61 itr = itr + 1;
    
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

to go to 0. Let me see this, for column equals 1 colon m b and

(Refer Slide Time: 40:59)



```
52 parity = prod(S);
53 treg(1:ti,il) = min1; %absolute value for all
54 treg(pos,il) = min2; %absolute value for min1 position
55 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61 end
62 end
63 itr = itr + 1;
64 end
%Counting errors
Nerrs = sum(msg ~= msg_cap);
```

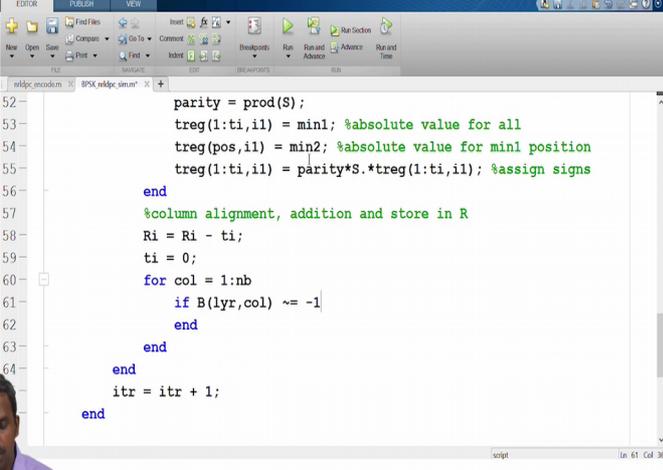


PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

if B of layer comma col not equal to minus 1, then you can

(Refer Slide Time: 41:09)



```
52 parity = prod(S);
53 treg(1:ti,il) = min1; %absolute value for all
54 treg(pos,il) = min2; %absolute value for min1 position
55 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61 if B(lyr,col) ~= -1
62 end
63 end
64 end
itr = itr + 1;
end
```



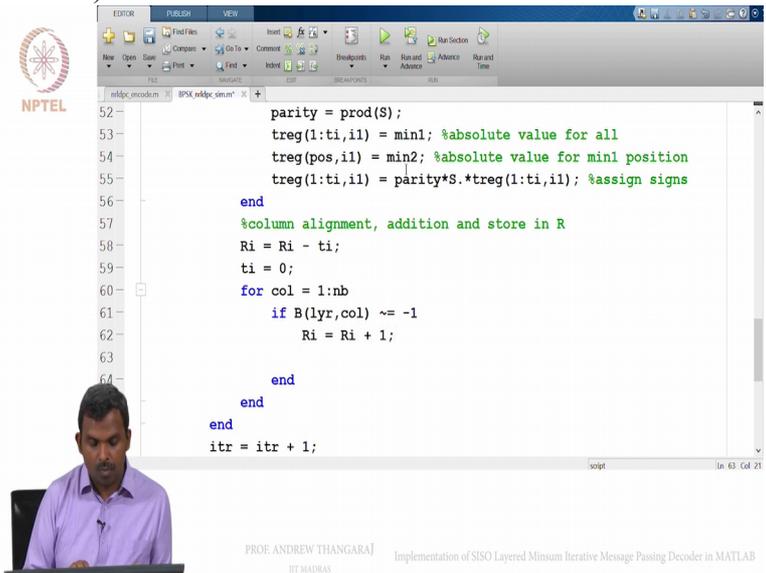
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

do something here.

So what do we do? I need to align according to the column and add it to L, right. So that is the, that is the important part there. So R of, R i equals R i plus 1.

(Refer Slide Time: 41:26)



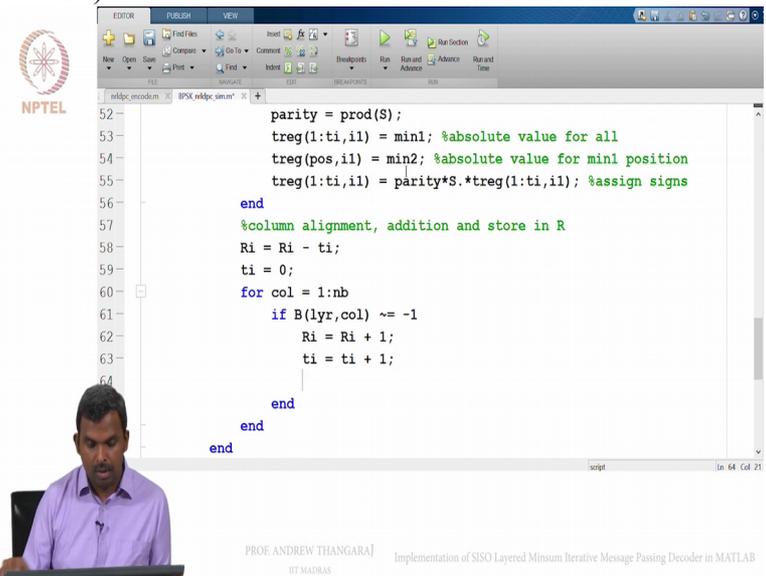
```
52 parity = prod(S);
53 treg(1:ti,il) = min1; %absolute value for all
54 treg(pos,il) = min2; %absolute value for min1 position
55 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63     end
64 end
65 end
66 itr = itr + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

t i equals t i plus 1.

(Refer Slide Time: 41:31)



```
52 parity = prod(S);
53 treg(1:ti,il) = min1; %absolute value for all
54 treg(pos,il) = min2; %absolute value for min1 position
55 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63         ti = ti + 1;
64     end
65 end
66 end
```

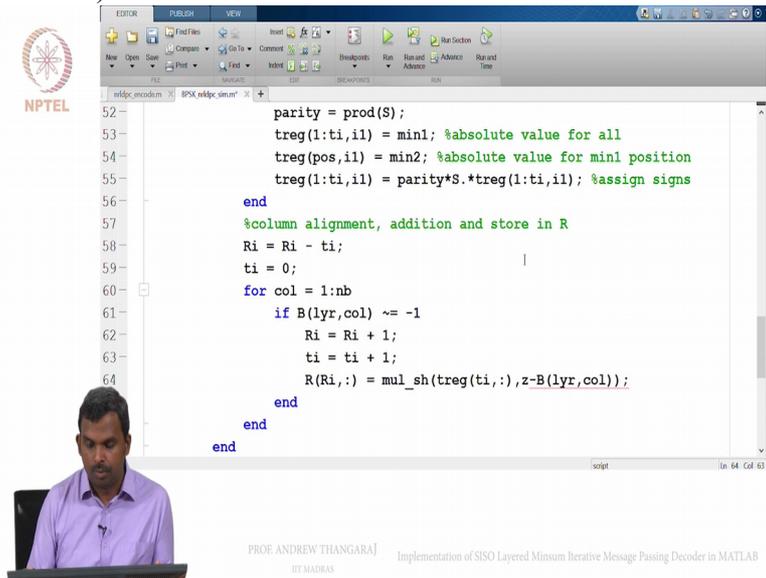
PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Then I store back. So R of R i comma colon equals, so the, so, so the alignment needs to change here again. So I pushed it, I did the subtraction correctly and then I shifted it so that I got the row alignment done.

Now I have to do the inverse of the shift. So try to think about that. So I have to reverse the shift and then align it and then store it in R, Ok. So R itself needs the alignment. So I need to mul shift of R of, of sorry t reg of t i comma colon, right with z minus B of l y r comma c o l. Ok

(Refer Slide Time: 42:30)



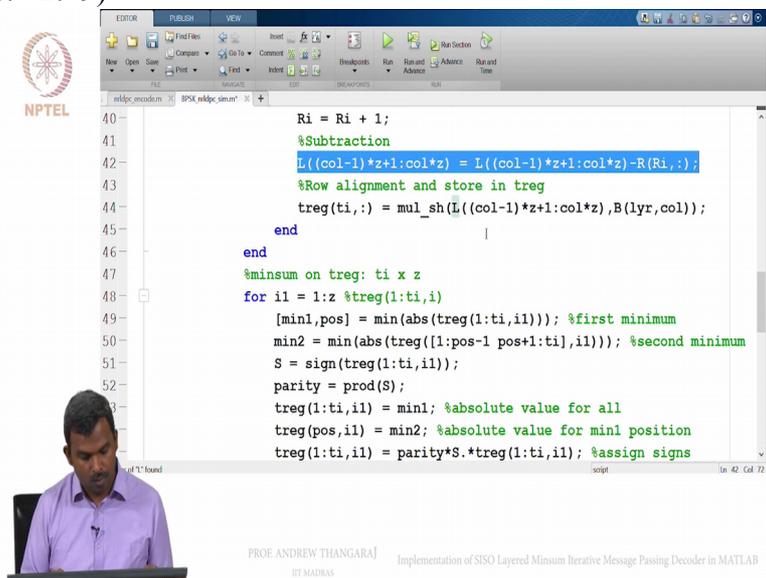
```
52 parity = prod(S);
53 treg(1:ti,il) = min1; %absolute value for all
54 treg(pos,il) = min2; %absolute value for min1 position
55 treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63         ti = ti + 1;
64         R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
65     end
66 end
67 end
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

so this is the inverse of the rotation and that gets stored in R.

Once you have stored this in R the L process is quite Ok. So we can go and do a cut and paste here. We do not have to the whole typing.

(Refer Slide Time: 42:49)

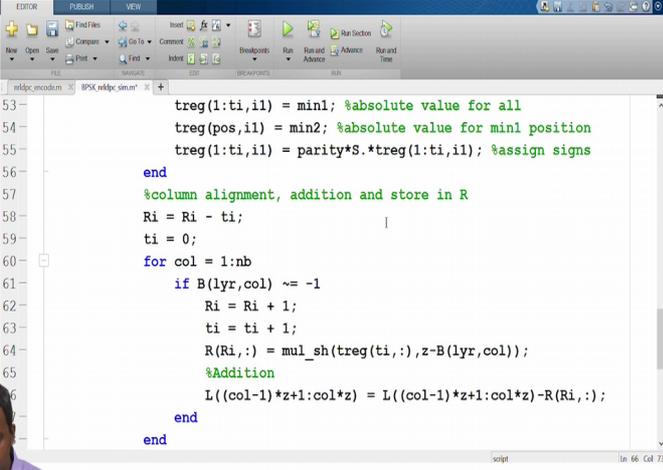


```
40 Ri = Ri + 1;
41 %Subtraction
42 L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43 %Row alignment and store in treg
44 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
45 end
46 end
47 %minsum on treg: ti x z
48 for il = 1:z %treg(1:ti,il)
49     [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
50     min2 = min(abs(treg(1:pos-1 pos+1:ti,il))); %second minimum
51     S = sign(treg(1:ti,il));
52     parity = prod(S);
53     treg(1:ti,il) = min1; %absolute value for all
54     treg(pos,il) = min2; %absolute value for min1 position
55     treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
```

PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Of course, this is the thing,

(Refer Slide Time: 42:59)



```
53     treg(1:ti,i1) = min1; %absolute value for all
54     treg(pos,i1) = min2; %absolute value for min1 position
55     treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63         ti = ti + 1;
64         R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
65         %Addition
66         L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
67     end
68 end
```

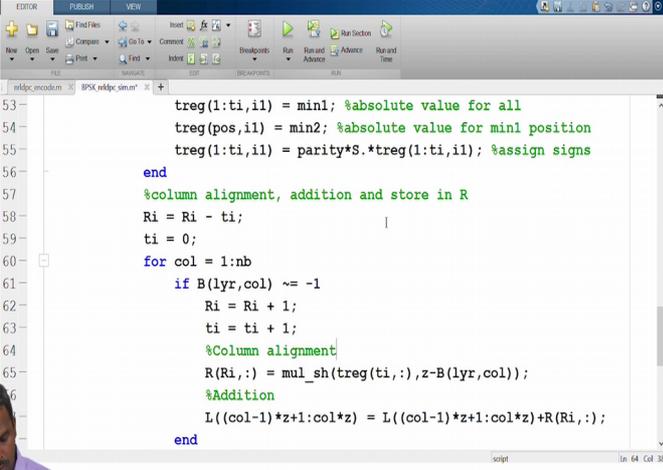
PROF. ANDREW THANGARAJ
IIT MADRAS
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok. So think I have come to the end of the iteration over every layer. I think I am doing everything here. We will try it out and see if I made any mistakes here.

So the total belief is stored there. Your iteration number and you have R i which counts the storage and then I go through and do the subtraction and then the alignment I store in the temporary thing. I then run through minsum on the temporary thing and then I column align again

But remember I have to do this adjustment on R i. I have to go back to the original index in my storage. And then store it back. I think I am doing Ok here, Ok. So this looks fine to me. I have done the column alignment here,

(Refer Slide Time: 43:56)



```
53 treg(1:ti,i1) = min1; %absolute value for all
54 treg(pos,i1) = min2; %absolute value for min1 position
55 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti;
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63         ti = ti + 1;
64         %Column alignment
65         R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
66         %Addition
67         L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
68     end
69 end
```

PROF. ANDREW THANGARAJ
IIT MADRAS

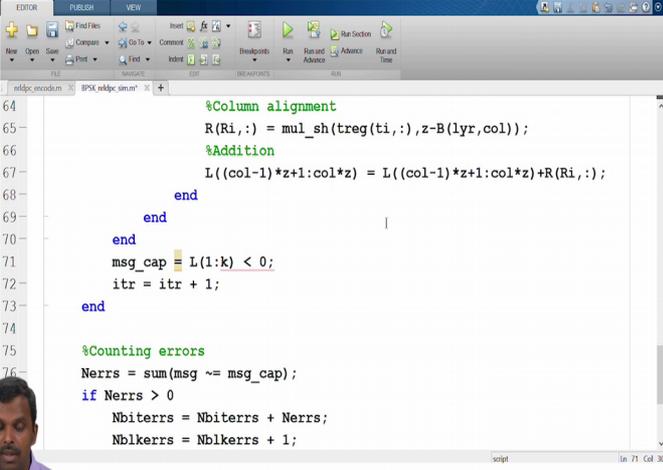
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok and then I have added it to L and we are good to continue.

So once I am done with this, I have done my, done my first layer. The layer is over. I have done all three operations on the layer. Once all the layers are over the iteration itself is complete, Ok. So now when the iteration is complete I have to do message cap, Ok. So I have to make decisions. So how we make decisions?

We know how to do this. message cap is 1 of 1 colon k, right the first k bits are the message bits and if they are less than

(Refer Slide Time: 44:37)



```
64 %Column alignment
65 R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
66 %Addition
67 L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
68 end
69 end
70 end
71 msg_cap = L(1:k) < 0;
72 itr = itr + 1;
73 end
74
75 %Counting errors
76 Nerrs = sum(msg ~= msg_cap);
if Nerrs > 0
    Nbiterrs = Nbiterrs + Nerrs;
    Nblkerrs = Nblkerrs + 1;
end
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

0, it is 1, if they are greater than 0, it is minus 1, Ok and that is the end of the iteration.

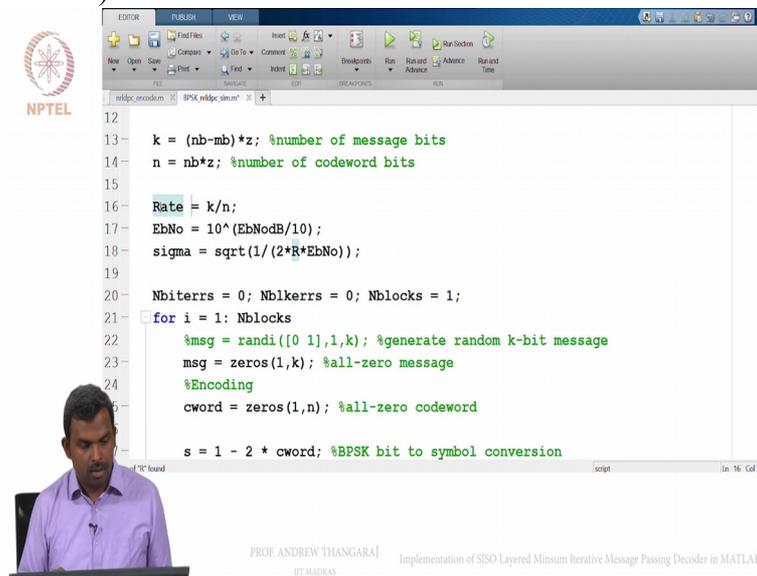
Once the iterations are done we come out and check if message is not equal to message cap, Ok. So that was the coding. Hopefully you saw the various things that I was doing. I am definitely not claiming that this is the most efficient way to code up your LDPC coder.

I am sure there are much better ways to code it. And if you are not interested in seeing how I was doing the coding you do not have to see it also. You can see the final code. It is available.

So let me just go through this code once just the way I have written it and we will, we will try it out. I mean I have to debug this. It should be another lecture but let me, let me show all the overall idea and how I have done that.

So you load the base matrix, take its size, and look at the expansion factor, number of non-minus 1s is important, that is the storage you need and then temporary storage. So there is some confusion about this R, Ok that is the rate, Ok. Can I redefine this as rate? Ok I will change this R as

(Refer Slide Time: 45:49)



```
12 k = (nb-mb)*z; %number of message bits
13 n = nb*z; %number of codeword bits
14
15
16 Rate = k/n;
17 EbNo = 10^(EbNodB/10);
18 sigma = sqrt(1/(2*R*EbNo));
19
20 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
21 for i = 1: Nblocks
22     msg = randi([0 1],1,k); %generate random k-bit message
23     msg = zeros(1,k); %all-zero message
24     %Encoding
25     cword = zeros(1,n); %all-zero codeword
26
27     s = 1 - 2 * cword; %BPSK bit to symbol conversion
```

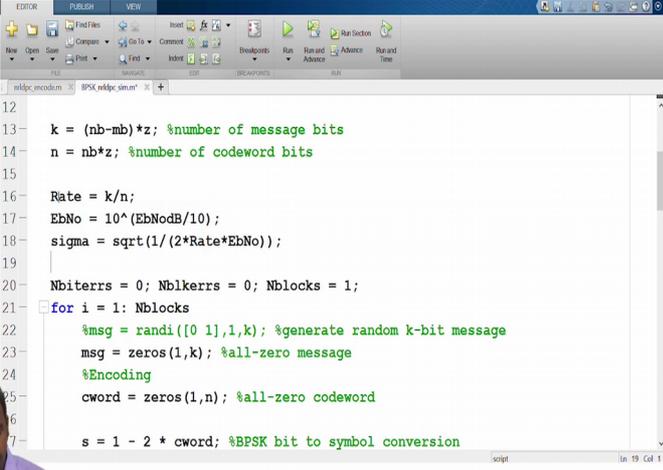
NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Rate here.

(Refer Slide Time: 45:54)



The screenshot shows a MATLAB script with the following code:

```
12
13 k = (nb-mb)*z; %number of message bits
14 n = nb*z; %number of codeword bits
15
16 Rate = k/n;
17 EbNo = 10^(EbNodB/10);
18 sigma = sqrt(1/(2*Rate*EbNo));
19
20 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 1;
21 for i = 1: Nblocks
22     %msg = randi([0 1],1,k); %generate random k-bit message
23     msg = zeros(1,k); %all-zero message
24     %Encoding
25     cword = zeros(1,n); %all-zero codeword
26
27     s = 1 - 2 * cword; %BPSK bit to symbol conversion
```

Below the code, a small video inset shows a man in a purple shirt. At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

So this Rate have any other role to play? Probably not, Ok so Rate is good.

So these are things to watch out for. If you reuse the variable MATLAB will warn you. That is a nice thing to have. So you can see that you are not making mistake here.

R is the storage for the row processing. t reg is the temporary storage k is the number of message bits, n b minus m b into z, n is the total number of codeword bits, Rate and you use Rate to find E b over N naught and sigma.

And then we are encoding the all zero codeword, so that is something, for simplicity we are doing that. For the decoder it is not so crucial. We do B P S K modulation as before and then add noise at standard deviation sigma.

Ok now I will start the soft decision iterative message passing layer decoding. You have your total belief L which is very important. Based on that you make decisions. Iteration number R i and I have outer loop for iterations, for every iteration you have to do this. We have to go through all the layers and for every layer you have to process all the columns, Ok.

So there are definitely much more efficient ways of doing these operations even in MATLAB but I am just doing it laboriously to show you how it works. You go through every column. Whenever you have a minus 1 you pull that value from the storage into the register but you

do this subtraction first, L is subtracted and then you do a row alignment and store in the temporary register.

So the row alignment once again, if you are confused about why I need to row alignment,

(Refer Slide Time: 47:26)

so if you see here, R is store

(Refer Slide Time: 47:29)

1 to 16 in this sequence. So this will be R of 1 comma 1, this will be

(Refer Slide Time: 47:36)

The slide content includes:

- NPTEL logo
- Handwritten notes: "1st row of Base matrix" with values 10, 5, 2, 15, -1. "Storage matrix" with a 16x16 identity matrix shifted right by 10 columns. "Total block" matrix structure:
$$\left[\begin{array}{c|c|c} \text{---} & \text{---} & \text{---} \\ \hline 16 & 16 & n_b \times 2 \end{array} \right]$$
- Video inset of Prof. Andrew Thangaraj, IIT Madras.
- Text at the bottom: "PROF. ANDREW THANGARAJ | Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB | IIT MADRAS"

R of 1 comma 2

(Refer Slide Time: 47:38)

This slide is identical to the one at 47:36, showing the same handwritten diagram and speaker video inset.

and so on.

Ok so this guy will be actually be, so this is 10, so this will actually be in MATLAB's thing, it will be R of 1 comma 11.

(Refer Slide Time: 47:46)

NPTEL

1st row of Base matrix

Storage matrix

16x16 identity shifted right by 10 columns

Total belief

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok. So R of 1 comma 11 actually corresponds to the first row after expansion. So R 1 comma 1 is not the first row after expansion. After expansion R of 1 comma 11 is the first row. So I have to align it so the first row matches throughout. So I can just go through .

So that is the alignment operation here.

(Refer Slide Time: 48:03)

NPTEL

```

31 L = r; %total belief
32 itr = 0; %iteration number
33 Ri = 0;
34 while itr < MaxItrs
35     for lyr = 1:nb
36         ti = 0;
37         for col = 1:nb
38             if B(lyr,col) ~= -1
39                 ti = ti + 1;
40                 Ri = Ri + 1;
41                 %Subtraction
42                 L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43                 %Row alignment and store in treg
44                 treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
45             end
46         end
47     end
48 end

```

PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

So we can see we have done the shifting by whatever value of B to get my, get my correct value done, Ok. So, I am a little, yeah I think this is Ok. So subtracted, and that is the value I am storing into my storage register, Ok.

So that is good. And once I, once I did that I have to do minsum. Ok and minsum I just do on t reg and this t i I have kept count. This t i is the number of non-minus 1s in each row.

(Refer Slide Time: 48:47)

The screenshot shows a MATLAB script with the following code:

```

35- for lyr = 1:mb
36-     ti = 0; %number of non -1 in row=lyr
37-     for col = 1:nb
38-         if B(lyr,col) ~= -1
39-             ti = ti + 1;
40-             Ri = Ri + 1;
41-             %Subtraction
42-             L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
43-             %Row alignment and store in treg
44-             treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
45-         end
46-     end
47-     %minsum on treg: ti x z
48-     for il = 1:z %treg(1:ti,il)
49-         [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
50-         min2 = min(abs(treg([1:pos-1 pos+1:ti],il))); %second minimum

```

Below the code, there is a small video inset of a man in a purple shirt speaking. At the bottom of the slide, it says "PROF. ANDREW THANGARAJ IIT MADRAS Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

Ok and so when row is l y r; t i is track of the number of non-minus 1s. So only the t i by z matter in t reg, there can be 19 rows, non minus 1 entries in each row at most so I am keeping it as t i by z, Ok.

And then I have this i 1. So I am processing the i 1th expanded row, there are z expanded rows in every block row, and the i 1th row I am processing one row after the other I am going.

I find the first minimum, absolute value right in the i 1th column, then the second minimum in the absolute value; I also need the position of the first minimum. And leave out the first minimum position and find the second minimum, right? So that is what I do here.

And then all the signs and then the product of the signs, the parity Ok. And then I assign the absolute value first after the minsum processing. Everything is min 1 except for the pos which will be the position of min 1 which gets replaced by min 2. This is just an efficient way of doing it very quickly.

And then, the sign is a sign, right. You have to multiply with the sign. I used the dot into here because both of these are vectors, you have to do dot into, Ok. And then we are ready to write back, Ok. So for that you need column alignment again.

So you remember t reg is row aligned and I have to make it column aligned, Ok. And when I have to make it column aligned I have to do the reverse of the row alignment rotation that I did, Ok.

So I have the other, so I am going back. This is reset the storage counter, Ok.

(Refer Slide Time: 50:28)

The screenshot shows a MATLAB editor window with the following code:

```

54 treg(pos,i1) = min2; %absolute value for min1 position
55 treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
56 end
57 %column alignment, addition and store in R
58 Ri = Ri - ti; %reset the storage counter
59 ti = 0;
60 for col = 1:nb
61     if B(lyr,col) ~= -1
62         Ri = Ri + 1;
63         ti = ti + 1;
64         %Column alignment
65         R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
66         %Addition
67         L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
68     end
69 end

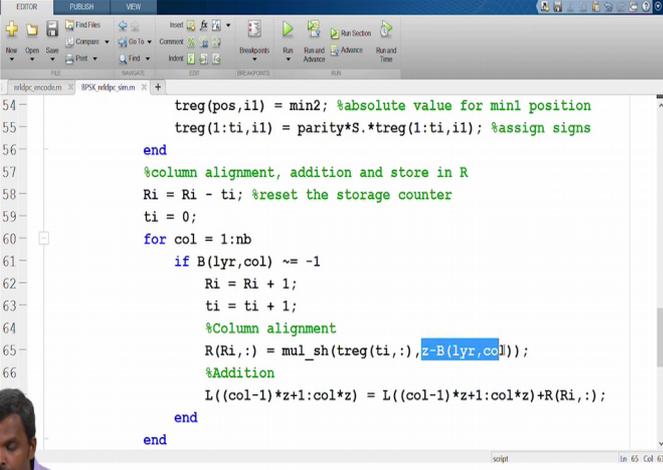
```

Below the code, a small video inset shows a man in a purple shirt. At the bottom of the slide, the text reads: "PROF ANDREW THANGARAJ, IIT MADRAS, Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB".

So we have come through, to the end of the row. You have to go back, reset it and then do that. Then reset the counter for the non-minus 1 entries and then go through the increment R i by 1, and this is my column alignment shift.

So you see instead of B of layer comma col, I am doing z minus B of layer comma col, right. So that is important to note,

(Refer Slide Time: 50:49)



PROF. ANDREW THANGARAJ
IIT MADRAS

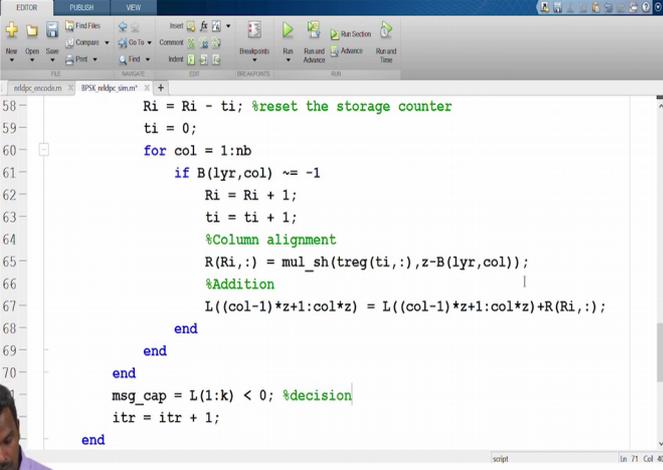
Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok. So this is t reg of t i comma colon and then I add, Ok. So I take the old values such that, L and then I add it to R of R i comma i.

So I think this is correct. Think I have done everything correctly but when we start debugging we will know if I made any mistake, minor error or what. May be you caught some minor error I made. Once we finish debugging we will have our code ready, Ok.

So remember total belief is the actual final belief you have based on which you have to make a decision. And that is the decision here. Ok so it is easy to do.

(Refer Slide Time: 51:24)



PROF. ANDREW THANGARAJ
IIT MADRAS

Implementation of SISO Layered Minsum Iterative Message Passing Decoder in MATLAB

Ok.

So lot of people will design on the entire codeword and use the parity check condition to see if the codeword is valid or not and exit earlier than the maximum number of iterations. I have not done that here. I am just showing you the whole iterations.

May be if you want you can add that later on. So that is why there is a while loop here, Ok. So once you come out, after that everything is the same, right what you did for the Hamming code, you have to do here. Count the total number of errors.

If there is any error at all, you increment the block errors by 1. And the bit errors get added on to nbit errors. Find the bitrate, frame error rate and divide. Nothing changes there, Ok

So hopefully I mean this is a little bit of a code. I mean, it is not very easy. It is not that complicated either, right. So the storage is what you have to manage carefully and, believe me this is not the only way to do it.

There are very, very efficient ways to write this code. In fact you can write this code without any for loops, Ok. So you can write it very efficiently in MATLAB using MATLAB's sparse matrix structure. It is possible to do these things.

But I am not doing all that right now. So I am just, this is more pedantic in the sense. I am just trying to explain how the operations go and focus on the operations as opposed to efficiency in code, Ok.

But still we have not debugged this yet. We will have to debug it and see how it works. We will do that in the next lecture. Thank you very much.