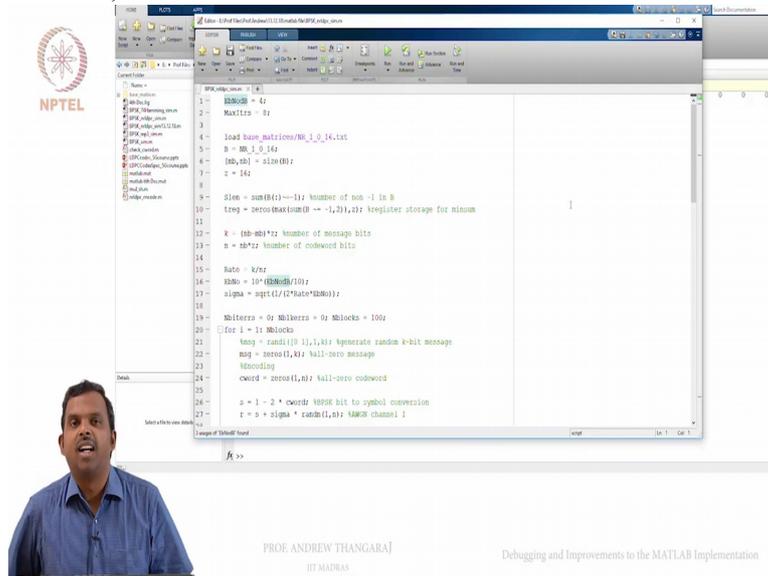


**LDPC and Polar codes in 5G Standard**  
**Professor Andrew Thangaraj**  
**Department of Electrical Engineering**  
**Indian Institute of Technology Madras**  
**Debugging and Improvements to the MATLAB Implementation**

(Refer Slide Time: 00:15)



```
1-- N=1024;
2-- MaxIter = 8;
3--
4-- load base_matrices/NB_1_0_16.tst
5-- B = NB_1_0_16;
6-- [nb,nb1] = size(B);
7-- z = 16;
8--
9-- Slen = sum(B(:)~=1); number of non-1 in B
10-- treg = zeros(max(Slen, z)-1, 2^z); register storage for minsum
11--
12-- k = (nb-nb1)*z; number of message bits
13-- n = nb*z; number of codeword bits
14--
15-- Rate = k/n;
16-- HSN0 = 10*(z/nb1)/10;
17-- sigma = sqrt(1/(2^Rate*HSN0));
18--
19-- MbIterns = 0; MbLockns = 0; MbLocks = 100;
20-- for i = 1: MbLocks
21--     msg = randi(0,1,1,8); generate random k-bit message
22--     msg = zeros(1,k); half-zero message
23--     uncoding
24--     cword = zeros(1,n); half-zero codeword
25--
26--     s = 1 - 2 * cword; 1024P bit to symbol conversion
27--     r = s + sigma * randn(1,n); AWGN channel 1
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

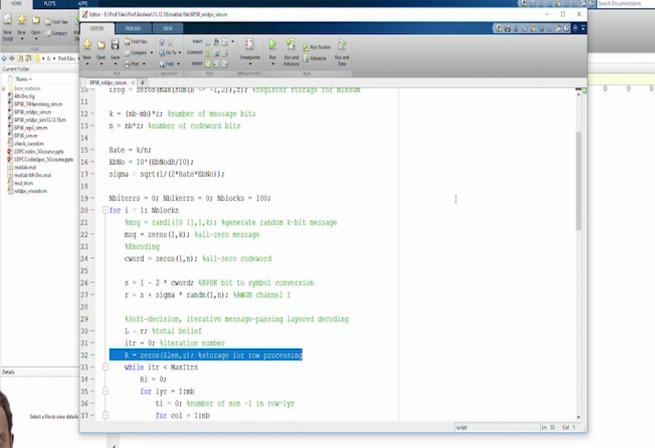
Debugging and Improvements to the MATLAB Implementation

Hello. Welcome to this lecture on LDPC decoding. In the previous lecture we did some MATLAB coding and we coded the LDPC message passing decoder.

If you remember we did not fully debug it. We mostly write and then I did some initial debugging. I will show you some of the changes I made. And then maybe we will make a few changes and run it and see how it works, Ok. So that is going to be the agenda for this class, Ok.

So if you look at this code, I made a few changes. The first change was at this line; at line 10 I had the initialization for the storage matrix for the row processing. That needs to actually move inside the block's,

(Refer Slide Time: 00:59)



```
11 R = zeros(256,256); % register storage for matrix
12 k = (26-26)/2; % number of message bits
13 n = nb/2; % number of codeword bits
14
15 Rate = k/n;
16 B2No = 10*(256-26)/10;
17 sigma = sqrt(1/(2*sigma*B2No));
18
19 Nbltters = 0; Nbltters = 0; Nbltters = 100;
20 for i = 1:Nblocks
21     msg = randi(0,1,1,1,1); %generate random k-bit message
22     msg = zeros(1,k); %all-zero message
23     %encoding
24     cword = zeros(1,n); %all-zero codeword
25
26     s = 1 - 2 * cword; %BPSK bit to symbol conversion
27     r = s + sigma * randi(1,n); %AWGN channel 1
28
29     %bit-decision, iterative message-passing layered decoding
30     L = rr %total belief
31     itr = 0; %iteration number
32     R = zeros(256,256); %storage for row processing
33     while itr < MaxIttr
34         Rl = 0;
35         for lyr = 1:nb
36             t1 = 0; % number of non -1 in row-lyr
37             for col = 1:nb
38                 %subtraction
39                 R((col-1)*n+1:col*2) = L((col-1)*n+1:col*2) - R(Rl,1);
```

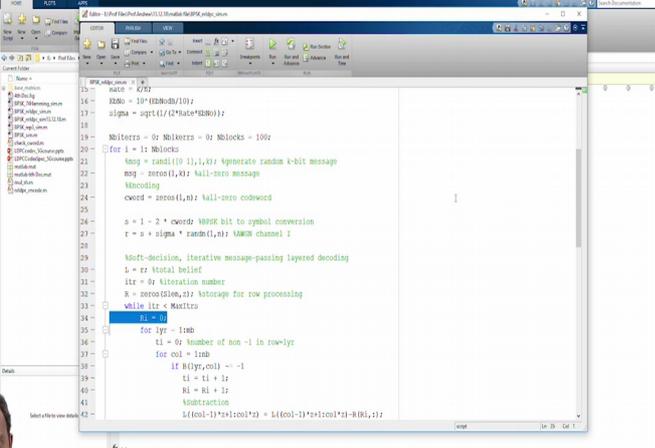
PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

block's loop. Because for every block you have to reset this R to 0, Ok. So that I moved inside this for i equal to 1 to n blocks.

And the next thing is this R i,

(Refer Slide Time: 01:10)



```
15 Rate = k/n;
16 B2No = 10*(256-26)/10;
17 sigma = sqrt(1/(2*sigma*B2No));
18
19 Nbltters = 0; Nbltters = 0; Nbltters = 100;
20 for i = 1:Nblocks
21     msg = randi(0,1,1,1,1); %generate random k-bit message
22     msg = zeros(1,k); %all-zero message
23     %encoding
24     cword = zeros(1,n); %all-zero codeword
25
26     s = 1 - 2 * cword; %BPSK bit to symbol conversion
27     r = s + sigma * randi(1,n); %AWGN channel 1
28
29     %bit-decision, iterative message-passing layered decoding
30     L = rr %total belief
31     itr = 0; %iteration number
32     R = zeros(256,256); %storage for row processing
33     while itr < MaxIttr
34         Rl = 0;
35         for lyr = 1:nb
36             t1 = 0; % number of non -1 in row-lyr
37             for col = 1:nb
38                 %subtraction
39                 R((col-1)*n+1:col*2) = L((col-1)*n+1:col*2) - R(Rl,1);
```

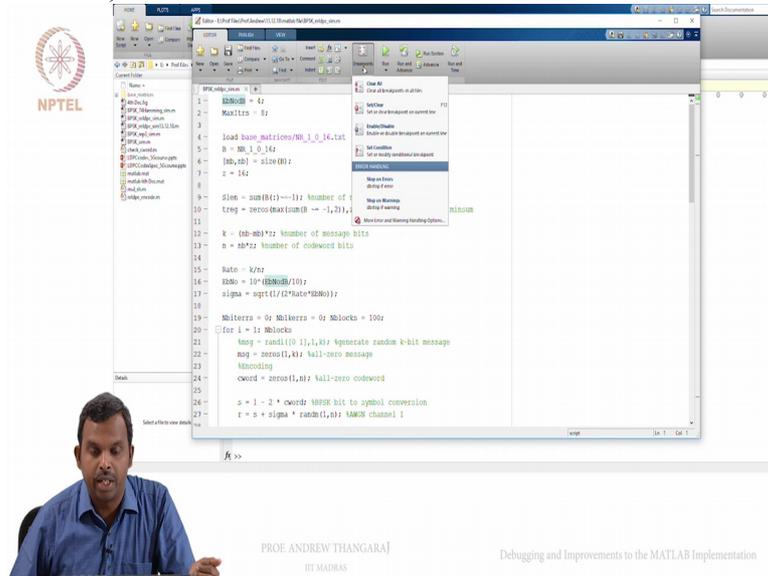
PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

R i needs to be initialized to 0 at the top of every iteration. So we start a new iteration, it has to start with R i 0, Ok. So that also I moved inside the while loop. These are the two changes I made.

With these two changes this code pretty much seems to be working. It is giving the expected results. So what I am going to do is first run through one step by step, I will run through this code and show you how, how the whole thing is working, Ok. So let us put

(Refer Slide Time: 01:39)



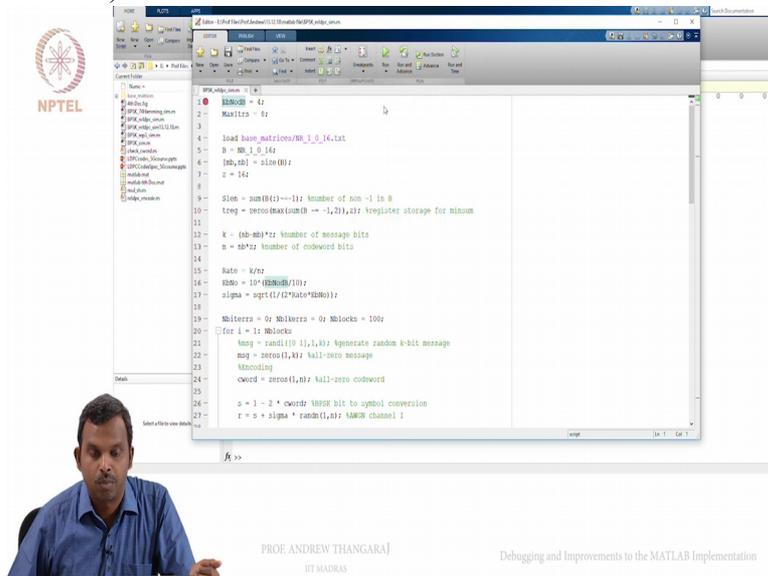
The screenshot shows the MATLAB code editor with the following code:

```
1: M=100; % 100
2: MaxIter = 5;
3:
4: load base_matrices/NO_1_0_14.txt
5: B = NO_1_0_14;
6: [nb,nb] = size(B);
7: z = 14;
8:
9: Slen = sum(B(:)-1); number of 1 in B
10: treg = zeros(max(Slen)-1,2); % register storage for minsum
11:
12: k = (nb-nb')/2; number of message bits
13: n = nb'*2; number of codeword bits
14:
15: Rate = k/n;
16: RSNR = 10*(20*log10(10));
17: sigma = sqrt(1/(2*Rate*RSNR));
18:
19: Mbitters = 0; Mberrors = 0; Mblocks = 100;
20: for i = 1: Mblocks
21:     msg = randi(0,1,1,k); %generate random k-bit message
22:     msg = zeros(1,k); %all-zero message
23:     %encoding
24:     cword = zeros(1,n); %all-zero codeword
25:
26:     s = 1 - 2 * cword; %BPSK bit to symbol conversion
27:     r = s + sigma * randn(1,n); %AWGN channel !
```

NPTEL  
PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

breakpoint here,

(Refer Slide Time: 01:41)



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

```
1: M=100; % 100
2: MaxIter = 5;
3:
4: load base_matrices/NO_1_0_14.txt
5: B = NO_1_0_14;
6: [nb,nb] = size(B);
7: z = 14;
8:
9: Slen = sum(B(:)-1); number of 1 in B
10: treg = zeros(max(Slen)-1,2); %register storage for minsum
11:
12: k = (nb-nb')/2; number of message bits
13: n = nb'*2; number of codeword bits
14:
15: Rate = k/n;
16: RSNR = 10*(20*log10(10));
17: sigma = sqrt(1/(2*Rate*RSNR));
18:
19: Mbitters = 0; Mberrors = 0; Mblocks = 100;
20: for i = 1: Mblocks
21:     msg = randi(0,1,1,k); %generate random k-bit message
22:     msg = zeros(1,k); %all-zero message
23:     %encoding
24:     cword = zeros(1,n); %all-zero codeword
25:
26:     s = 1 - 2 * cword; %BPSK bit to symbol conversion
27:     r = s + sigma * randn(1,n); %AWGN channel !
```

NPTEL  
PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

Ok.

So this is something important. You will single step-through the code to see how it works. And then if you run it, your debugger stopped it there and then you can start single stepping, Ok.

So this is loading base matrix, assigning it to this, we do not expect too many errors here. This is the number of non minus 1s in B and then you assign a t reg, and then k and n, Rate, E b over N naught sigma, initialize k into the loop Ok.

So this is message codeword and then the symbol r, L equals r, is being initialized and we start the recursion, Ok, start the decoder.

Ok

(Refer Slide Time: 02:20)

The screenshot shows a MATLAB command window with the following code and output:

```

Columns 346 through 352
0 0 0 0 0 0 0
E>> B1
B1 =
314
>> BPSK_nrlqpc_sim
4 0 0 0 0 1
>> BPSK_nrlqpc_sim
1.0e+05 *
0.0000 0.0000 0.0000 0.0100 1.4010 0.0100
>> format short g
>> disp([BPSK_nrlqpc_sim BER_sim MbitRate MbitRate MbitRate])
4 0.599 0.39024 999 1.0010e+05 1000
>> BPSK_nrlqpc_sim
4 0 0 0 0 10
>> BPSK_nrlqpc_sim
4 0 0 0 0 100
>> BPSK_nrlqpc_sim
4 0 0 0 0 100
  
```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, with the text "PROF ANDREW THANGARAJ, IIT MADRAS" and "Debugging and Improvements to the MATLAB Implementation".

so let us see a few values. May be you can see L of 1 colon 32, the first two

(Refer Slide Time: 02:25)

The screenshot shows a MATLAB command window with the following code and output:

```

>> disp([BPSK_nrlqpc_sim BER_sim MbitRate MbitRate MbitRate])
4 0.599 0.39024 999 1.0010e+05 1000
>> BPSK_nrlqpc_sim
4 0 0 0 0 10
>> BPSK_nrlqpc_sim
4 0 0 0 0 100
>> BPSK_nrlqpc_sim
4 0 0 0 0 100
>> BPSK_nrlqpc_sim
E>> L(1:32)
ans =
Columns 1 through 10
0.46519 0.46291 -1.1391 2.3932 -2.1616 1.2325 0.70626 -0.29445 1.4444 2.4122
Columns 11 through 20
0.42474 2.2956 1.1442 0.52215 -0.21567 1.7113 0.44536 1.7010 2.0553 2.2162
Columns 21 through 30
1.4070 0.40702 1.9742 0.39066 0.54767 1.0575 0.45455 2.9797 1.3249 0.17
Columns 31 through 32
0.85479 1.7259
  
```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, with the text "PROF ANDREW THANGARAJ, IIT MADRAS" and "Debugging and Improvements to the MATLAB Implementation".



(Refer Slide Time: 02:54)

NPTEL

```

Columns 11 through 20
0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44536    1.7818    2.0553    2.2142

Columns 21 through 30
1.4078    0.40702    1.9742    0.39666    0.94767    1.0575    0.05459    2.9797    1.3249    0.17

Columns 31 through 32
0.05479    1.7259

E>> sigma
sigma =
    0.78430

E>> k
k =
    352

E>> n
n =
    1088

E>> Rate
Rate =
    0.32953
    
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

n is 1 0 8 8, Ok.

So the code rate, if you look at it, I think I had a Rate here which is

(Refer Slide Time: 03:04)

NPTEL

```

Columns 31 through 32
0.05479    1.7259

E>> sigma
sigma =
    0.78430

E>> k
k =
    352

E>> n
n =
    1088

E>> Rate
Rate =
    0.32953
    
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

point 3, Ok. So it is like one third roughly one third, slightly smaller than that for some reason. So we will see why, why that is so. Because that are 46 by 68 right so this is slightly smaller than one third.

So  $E_b/N_0$  of 4 dB and all is quite high  $E_b/N_0$ , but still the noise will be quite low, quite high. Sigma is point 7 8 4 8.

(Refer Slide Time: 03:29)

The screenshot shows the MATLAB interface. The Command Window displays the following text:

```

Columns 31 through 32
    1.4870    0.48702    1.9742    0.39944    0.94767    1.0575    0.05459    2.9797    1.3249    0.17

E>> sigma
sigma =
    0.78433

E>> k
k =
    352

E>> n
n =
    1088

E>> Rate
Rate =
    0.32353

E>>
  
```

At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Debugging and Improvements to the MATLAB Implementation".

So you can expect quite high noise. And you can see, so many of the

(Refer Slide Time: 03:34)

The screenshot shows the MATLAB interface. The Command Window displays the following text:

```

E>> BER('21212121')
ans =
Columns 1 through 10
    0.44519    0.45291    0.78433    2.3932    -2.1616    1.2325    0.79626    -0.29445    1.4444    2.4122

Columns 11 through 20
    0.42474    2.2954    1.1442    0.52215    -0.21547    1.7113    0.44536    1.7818    2.0553    2.2142

Columns 21 through 30
    1.4870    0.48702    1.9742    0.39944    0.94767    1.0575    0.05459    2.9797    1.3249    0.17

Columns 31 through 32
    0.85479    1.7259

E>> sigma
sigma =
    0.78433

E>> k
k =
  
```

At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ, IIT MADRAS, Debugging and Improvements to the MATLAB Implementation".

received words are in error.

Ok, if you look at this minus 2,

(Refer Slide Time: 03:37)

NPTEL

```

>> BFGF_stridge_min
ED> 1(1:32)
ans =
Columns 1 through 10
    0.44519    0.45291   -1.1391    2.3932    3.3116    1.2325    0.70626   -0.29445    1.4444    2.4122
Columns 11 through 20
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44536    1.7818    2.0553    2.2142
Columns 21 through 30
    1.4978    0.49702    1.9742    0.39664    0.94767    1.0575    0.05459    2.9797    1.3249    0.17
Columns 31 through 32
    0.85479    1.7259
ED> sigma
sigma =
    0.78438
ED> k
k =
    1
  
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok so that is lot of error, minus point 2,

(Refer Slide Time: 03:39)

NPTEL

```

>> BFGF_stridge_min
ED> 1(1:32)
ans =
Columns 1 through 10
    0.44519    0.45291   -1.1391    2.3932   -2.1616    1.2325    0.70626   -0.29445    1.4444    2.4122
Columns 11 through 20
    0.42474    2.2956    1.1442    0.52215   10.2133    1.7113    0.44536    1.7818    2.0553    2.2142
Columns 21 through 30
    1.4978    0.49702    1.9742    0.39664    0.94767    1.0575    0.05459    2.9797    1.3249    0.17
Columns 31 through 32
    0.85479    1.7259
ED> sigma
sigma =
    0.78438
ED> k
k =
    1
  
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

minus point 9,



(Refer Slide Time: 03:56)

```
17 - sigma = sqrt(L*(2*Rate)*ERBN0);
18 - Nblocks = 0; Nblocks = 0; Nblocks = 100;
19 -
20 - for i = 1: Nblocks
21 -     msg = randi(0:1,1,K); %generate random k-bit message
22 -     msg = zeros(1,K); %all-zero message
23 -     %encoding
24 -     word = zeros(1,K); %all-zero codeword
25 -
26 -     s = 1 - 2 * (word && 0x08 bit to symbol conversion
27 -     z = s + sigma * randn(1,N); %AWGN channel 1
28 -
29 -     %Soft-decision, iterative message-passing layered decoding
30 -     L = tr %total belief
31 -     itr = 0; %iteration number
32 -     h = zeros(1,N); %storage for row processing
33 -     while itr <= Nblocks
34 -         K1 = 0;
35 -         for ltr = 1:nb
36 -             %l = 0; number of non-1 in row=ltr
37 -             for col = 1:nb
38 -                 if B(ltr,col) == -1
39 -                     %l = l + 1;
40 -                     K1 = K1 + 1;
41 -                 %subtraction
42 -                 L((col-1)*n+1:col*n) = L((col-1)*n+1:col*n) - R(ltr,l);
43 -                 %row alignment and store in treg
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 -
```

hopefully corrected so you can see here I am going through the row processing each layer or each block row in the, in the base matrix and for every block row I am going to try and do the minsum processing but first thing is you have to do the subtraction.

First loop does the subtraction part and row alignment and storing it and this t reg, Ok

(Refer Slide Time: 04:21)

```
17 - sigma = sqrt(L*(2*Rate)*ERBN0);
18 - Nblocks = 0; Nblocks = 0; Nblocks = 100;
19 -
20 - for i = 1: Nblocks
21 -     msg = randi(0:1,1,K); %generate random k-bit message
22 -     msg = zeros(1,K); %all-zero message
23 -     %encoding
24 -     word = zeros(1,K); %all-zero codeword
25 -
26 -     s = 1 - 2 * (word && 0x08 bit to symbol conversion
27 -     z = s + sigma * randn(1,N); %AWGN channel 1
28 -
29 -     %Soft-decision, iterative message-passing layered decoding
30 -     L = tr %total belief
31 -     itr = 0; %iteration number
32 -     h = zeros(1,N); %storage for row processing
33 -     while itr <= Nblocks
34 -         K1 = 0;
35 -         for ltr = 1:nb
36 -             %l = 0; number of non-1 in row=ltr
37 -             for col = 1:nb
38 -                 if B(ltr,col) == -1
39 -                     %l = l + 1;
40 -                     K1 = K1 + 1;
41 -                 %subtraction
42 -                 L((col-1)*n+1:col*n) = L((col-1)*n+1:col*n) - R(ltr,l);
43 -                 %row alignment and store in treg
44 -                 treg(ltr,l) = min(L((col-1)*n+1:col*n), B(ltr,col));
45 -             end
46 -         end
47 -         %minsum on treg; ltr = 1
48 -         for ll = 1:itr; treg(ll,l);
49 -             [minl,pos] = minabs(treg(ll,l)); %first minsum
50 -         end
51 -         itr = itr + 1;
52 -     end
53 - end
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 -
```

so if remember that is what it does.

So, so if you let it run through for the first layer, if you let it run and run to the cursor, if you can look at this t reg,

(Refer Slide Time: 04:33)

The screenshot shows the MATLAB workspace containing a matrix with 19 rows and 16 columns. The presenter, Pro Andrew Thangaraj, is visible in the bottom left corner.

Variable	Value
ans	0.26453 -0.041427 -1.3471 -0.29701 0.8104 0.26560 0.089643 0.42117 0.38727 1.4975
BlockA	-0.4956 0.0217 1.2518 0.75150 0.84764 1.6465 1.7043 0.47120 -0.36560 1.2511
IPG_16mming_mom	0.30585 -0.095629 -0.040455 1.4143 1.8766 0.43901 1.4015 0.19991 -0.22577 0.75804
IPG_16mm_12.5mm	1.6937 0.88094 0.50460 1.6204 0.84470 1.1399 0.88390 0.48739 1.3770 0.69749
IPG_16mm	1.2053 0.82508 -0.020208 0.97512 1.4701 0.23703 -0.027401 1.3503 -0.48385 -0.28064
BlockA_mom	1.9362 1.463 1.0717 -0.93031 0.16194 0.89190 0.42000 0.47920 0.42502 0.4312
IPGCode_Strainmagh	1.9367 1.1374 1.1022 2.2093 1.814 1.4229 -0.052004 1.0571 2.3077 0.97415
BlockA_mom	-2.1416 1.2325 0.70426 -0.29445 1.4444 2.4122
IPG_16mm	1.7259 0.44536 1.7810 2.0553 2.2162 1.4970
IPGCode_Strainmagh	-0.41376 -0.38973 0.99359 0.49465 1.7741 0.43327
BlockA_mom	0.47606 -0.50580 1.2014 -0.83932 -0.22203 1.243
IPG_16mm	0.70549 1.4036 0.32382 -0.72864 0.35886 1.4392
BlockA_mom	0.78871 1.2862 2.442 0.38229 -1.275 -0.548
IPGCode_Strainmagh	0.16186 0.99737 1.3339 -0.44447 2.121 0.36480
BlockA_mom	0.93727 0.85451 -0.23433 0.44954 0.73113 0.51589
IPG_16mm	1.0961 1.061 1.9183 -0.033278 -0.12709 2.0744
BlockA_mom	1.401 1.2583 0.79507 0.42264 2.4749 0.5964
IPGCode_Strainmagh	1.4309 1.5069 1.0372 1.0768 0.58073 1.3015
BlockA_mom	-0.45334 0.74469 1.3591 1.5233 1.4926 1.3790
IPG_16mm	2.0597 0.43346 1.5976 1.5278 0.8149 1.3902
BlockA_mom	2.2471 -1.3078 0.60232 0.78807 1.1919 0.68163
IPGCode_Strainmagh	0.31543 0.42211 0.86425 1.8652 0.21847 1.4160
BlockA_mom	0.74144 1.9526 0.18023 0.53199 0.29206 0.58523
IPG_16mm	-0.1334 2.3011 1.3534 -0.18566 0.59092 1.4053
BlockA_mom	0.10857 1.8354 1.7335 1.3799 2.192 0.95602
IPGCode_Strainmagh	0.99914 0.90086 1.9472 2.1172 1.837 0.89678

it will give you an idea of what is going on. So it needs to have, if I am not wrong 19 rows, because there were 19 non minus 1s in the first block row, Ok and then each row will have 16 values.

These are the aligned values corresponding to the first thing. So for instance if you look at B of 1 comma 1,

(Refer Slide Time: 04:52)

The screenshot shows the MATLAB command window with the command `B(1,1)` entered, resulting in the output `10`. The presenter, Pro Andrew Thangaraj, is visible in the bottom left corner.

it is 10, Ok. Then if you look at L of 1 colon 16,

(Refer Slide Time: 04:56)

The screenshot shows the MATLAB interface. The workspace contains several variables, including a matrix 'B'. The command window shows the following operations:

```

ED> B(1,1)
ans =
    10
ED> B(1:10)
ans =
Columns 1 through 10
    0.44519    0.45291   -1.1391    2.3932   -2.1616    1.2325    0.70626   -0.29445    1.4444    2.4122
Columns 11 through 16
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113
  
```

Below the screenshot is a small video inset of Prof. Andrew Thangaraj, with the text "PROF. ANDREW THANGARAJ, IIT MADRAS" and "Debugging and Improvements to the MATLAB Implementation".

this is the actual received value column aligned Ok. Now this B of 1 comma 1 is 10, so you have to rotate it by 10 positions and if you look at the t reg of 1 comma colon it will be the same thing rotated by 10 positions.

So we will start

(Refer Slide Time: 05:12)

The screenshot shows the MATLAB interface. The command window shows the following operations:

```

ED> B(1,1)
ans =
    10
ED> B(1:10)
ans =
Columns 1 through 10
    0.44519    0.45291   -1.1391    2.3932   -2.1616    1.2325    0.70626   -0.29445    1.4444    2.4122
Columns 11 through 16
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113
ED> treg(1,1)
ans =
Columns 1 through 10
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44519    0.45291   -1.1391    2.3932
Columns 11 through 16
   -2.1616    1.2325    0.70626   -0.29445    1.4444    2.4122
  
```

Below the screenshot is a small video inset of Prof. Andrew Thangaraj, with the text "PROF. ANDREW THANGARAJ, IIT MADRAS" and "Debugging and Improvements to the MATLAB Implementation".

at the 11th column. So you can see this is what starts here,

(Refer Slide Time: 05:15)

The screenshot shows the MATLAB IDE with the following code and output:

```

ED> B(1,1)
ans =
10
ED> L(1:16)
ans =
Columns 1 through 10
0.44519    0.62291   -1.1391    2.3932   -2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
Columns 11 through 16
0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113
ED> treg(1,:)
ans =
Columns 1 through 10
0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44519    0.62291   -1.1391    2.3932
Columns 11 through 16
-2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
ED>
  
```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

point 4 2 4 7 4. So now if proceed, it starts at the eleventh column and then it is rotating. So this has happened. So this is t reg for you. Ok. Now if you proceed this loop does the minsum processing,

(Refer Slide Time: 05:26)

The screenshot shows the MATLAB IDE with the following code and output:

```

ED> B(1,1)
ans =
10
ED> L(1:16)
ans =
Columns 1 through 10
0.44519    0.62291   -1.1391    2.3932   -2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
Columns 11 through 16
0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113
ED> treg(1,:)
ans =
Columns 1 through 10
0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44519    0.62291   -1.1391    2.3932
Columns 11 through 16
-2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
ED>
  
```

ED> B(1,1)
li = li - 1;
for col = 1:nb
 if B(li,col) <= -1
 li = li + 1;
 ki = ki + 1;
 subtraction
 t((col-1)\*n+1:col) = li((col-1)\*n+1:col) - B(li,col);
 %row alignment and store in treg
 treg(li,col) = min(abs(treg(li,1:n)+1:col),B(li,col));
 end
end
Minimum on treg: li x z
for ii = 1:nrow(treg)
 [min1,pos] = min(abs(treg(ii,1:n))); %first minimum
 min2 = min(abs(treg(ii,pos+1:n))); %second minimum
 S = sign(treg(ii,1:n));
 parity = prod(S);
 treg(ii,1:n) = min1: absolute value for all
 treg(pos,1:n) = min2: absolute value for min1 position
 treg(ii,1:n) = parity\*S.\*treg(ii,1:n); %assign signs
end
%column alignment, addition and store in B
ki = ki - 1; %reset the storage counter
li = li;
for col = 1:nb
 if B(li,col) <= -1
 

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

right. And it works just on t reg, and if you fully finish the minsum, you expect the min 1 and min 2 to replace the value.

So if you

(Refer Slide Time: 05:35)

The screenshot shows the MATLAB interface. The workspace contains variables: `h`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `h7`, `h8`, `h9`, `h10`, `h11`, `h12`, `h13`, `h14`, `h15`, `h16`, `h17`, `h18`, `h19`, `h20`, `h21`, `h22`, `h23`, `h24`, `h25`, `h26`, `h27`, `h28`, `h29`, `h30`, `h31`, `h32`, `h33`, `h34`, `h35`, `h36`, `h37`, `h38`, `h39`, `h40`, `h41`, `h42`, `h43`, `h44`, `h45`, `h46`, `h47`, `h48`, `h49`, `h50`, `h51`, `h52`, `h53`, `h54`, `h55`, `h56`, `h57`, `h58`, `h59`, `h60`, `h61`, `h62`, `h63`, `h64`, `h65`, `h66`, `h67`, `h68`, `h69`, `h70`, `h71`, `h72`, `h73`, `h74`, `h75`, `h76`, `h77`, `h78`, `h79`, `h80`, `h81`, `h82`, `h83`, `h84`, `h85`, `h86`, `h87`, `h88`, `h89`, `h90`, `h91`, `h92`, `h93`, `h94`, `h95`, `h96`, `h97`, `h98`, `h99`, `h100`.

```

ED> h(1,1)
ans =
    10
ED> t(1:16)
ans =
Columns 1 through 10
    0.44519    0.45291   -1.1391    2.3932   -2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
Columns 11 through 16
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113
ED> treg(1,1)
ans =
Columns 1 through 10
    0.42474    2.2956    1.1442    0.52215   -0.21567    1.7113    0.44519    0.45291   -1.1391    2.3932
Columns 11 through 16
   -2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
ED>
    
```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

go through and look at t reg of 1 comma colon, so, so this is not, t reg of 1 comma colon is not where the minsum has run. On the other hand it is t reg of colon comma 1,

(Refer Slide Time: 05:45)

The screenshot shows the MATLAB interface. The workspace contains variables: `h`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `h7`, `h8`, `h9`, `h10`, `h11`, `h12`, `h13`, `h14`, `h15`, `h16`, `h17`, `h18`, `h19`, `h20`, `h21`, `h22`, `h23`, `h24`, `h25`, `h26`, `h27`, `h28`, `h29`, `h30`, `h31`, `h32`, `h33`, `h34`, `h35`, `h36`, `h37`, `h38`, `h39`, `h40`, `h41`, `h42`, `h43`, `h44`, `h45`, `h46`, `h47`, `h48`, `h49`, `h50`, `h51`, `h52`, `h53`, `h54`, `h55`, `h56`, `h57`, `h58`, `h59`, `h60`, `h61`, `h62`, `h63`, `h64`, `h65`, `h66`, `h67`, `h68`, `h69`, `h70`, `h71`, `h72`, `h73`, `h74`, `h75`, `h76`, `h77`, `h78`, `h79`, `h80`, `h81`, `h82`, `h83`, `h84`, `h85`, `h86`, `h87`, `h88`, `h89`, `h90`, `h91`, `h92`, `h93`, `h94`, `h95`, `h96`, `h97`, `h98`, `h99`, `h100`.

```

Columns 11 through 16
   -2.1616    1.2325    0.70426   -0.29445    1.4444    2.4122
ED> treg(1,1)
ans =
    0.42474
    0.48702
    1.2289
    2.2786
    1.4251
    0.45229
    2.0249
    0.49024
    1.5159
    1.2021
    1.0822
    0.46292
    0.54433
   -0.44956
    0.30585
    1.4857
    1.3953
    1.8942
    1.9387
ED>
    
```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

Ok the first column, remember. So I rotated the t reg, aligned it and then stored it as a row, ok. 16 values I stored as a row, so my first column in t reg is actually the first expanded row in the parity check matrix, Ok.

So I will look at the first column and on this the

(Refer Slide Time: 06:04)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

minsum is run, Ok. So to look at it a bit more cleanly may be I look at the transpose

(Refer Slide Time: 06:08)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

of this so that you can see it is a row. So we need to find min 1 and min 2 here, right.

(Refer Slide Time: 06:14)

NPTEL

Current Folder: ... \Prof.A... \Prof.A... \matlab\...

Current Workspace: ... \matlab\...

ans =

0.4529	0.48702	1.2288	2.2786	1.4251	0.4529	2.0249	0.89024	1.9159	1.2021
1.0822	0.92392	0.56433	-0.44956	0.30585	1.4937	1.2053	1.8362	1.9387	

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

So if you look at it, maybe we can eyeball min 1 and min 2, point 4 2 looks pretty low to me.

(Refer Slide Time: 06:21)

NPTEL

Current Folder: ... \Prof.A... \Prof.A... \matlab\...

Current Workspace: ... \matlab\...

ans =

0.4529	0.48702	1.2288	2.2786	1.4251	0.4529	2.0249	0.89024	1.9159	1.2021
1.0822	0.92392	0.56433	-0.44956	0.30585	1.4937	1.2053	1.8362	1.9387	

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

I do not think there is anything else lower than point 4 2, so yeah this should be min 1, min 1 should be point 4 2.

And min 2 looks like it is point 4 5,

(Refer Slide Time: 06:29)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Columns 1 through 10
0.42474 0.48702 1.2288 2.2786 1.4251 0.4529 2.0249 0.49024 1.9199 1.2021
Columns 11 through 19
1.0822 0.92392 0.56433 -0.44956 0.30585 1.4937 1.2053 1.4362 1.9387

Ok. So min 2 is point 4 5 and min 1 is point 4 2, number 1.

(Refer Slide Time: 06:38)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Columns 1 through 10
0.42474 0.48702 1.2288 2.2786 1.4251 0.4529 2.0249 0.49024 1.9199 1.2021
Columns 11 through 19
1.0822 0.92392 0.56433 -0.44956 0.30585 1.4937 1.2053 1.4362 1.9387

That is the first thing that minsum calculates. Next is the overall parity. The overall parity is actually minus 1, right. So there is only one minus 1 here. So if I multiply the signs, you will get minus 1.

So after minsum processing I am expecting point 4 2 4 7 4 to be all these values, all these other values will be point 4 2 4 7 4. This alone will be point 4 5 2 9 and then all the signs will be flipped. There will be only positive

(Refer Slide Time: 07:03)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

ans =	1.2288	2.2786	1.4251	0.4529	2.0249	0.39024	1.9199	1.2021	1.0822	0.52392	0.56433	0.30585	1.4937	1.2053	1.8362	1.9387		
Columns 1 through 10	0.42474	0.48702	1.2288	2.2786	1.4251	0.4529	2.0249	0.39024	1.9199	1.2021	1.0822	0.52392	0.56433	0.30585	1.4937	1.2053	1.8362	1.9387
Columns 11 through 19	1.0822	0.52392	0.56433	0.30585	1.4937	1.2053	1.8362	1.9387										

sign here; everything else will be negative sign. That is what I expect after the minsum. Let us see it happens.

So we run through the minsum, just for this guy then let us say step

(Refer Slide Time: 07:15)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

ans =	1.2288	2.2786	1.4251	0.4529	2.0249	0.39024	1.9199	1.2021	1.0822	0.52392	0.56433	-0.46956	0.30585	1.4937	1.2053	1.8362	1.9387	
Columns 1 through 10	0.42474	0.48702	1.2288	2.2786	1.4251	0.4529	2.0249	0.39024	1.9199	1.2021	1.0822	0.52392	0.56433	0.30585	1.4937	1.2053	1.8362	1.9387
Columns 11 through 19	1.0822	0.52392	0.56433	-0.46956	0.30585	1.4937	1.2053	1.8362	1.9387									

and we see the same thing, right.



(Refer Slide Time: 07:45)

The screenshot shows the MATLAB environment. The workspace contains several variables: `l` (0.30595), `l_0` (1.6933), `l_1` (1.2053), `l_2` (1.8362), `l_3` (1.9387), `l_4` (1.9387), `l_5` (1.9387), `l_6` (1.9387), `l_7` (1.9387), `l_8` (1.9387), `l_9` (1.9387), `l_10` (1.9387), `l_11` (1.9387), `l_12` (1.9387), `l_13` (1.9387), `l_14` (1.9387), `l_15` (1.9387), `l_16` (1.9387), `l_17` (1.9387), `l_18` (1.9387), `l_19` (1.9387). The command window shows the following operations:

```

E>> l = l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + l_8 + l_9 + l_10 + l_11 + l_12 + l_13 + l_14 + l_15 + l_16 + l_17 + l_18 + l_19
ans =
Columns 1 through 10
0.42474    0.48702    1.2288    2.2786    1.4251    0.4529    2.0249    0.89024    1.9159    1.2021
Columns 11 through 19
1.0822    0.92392    0.56433    -0.44956    0.30585    1.4937    1.2053    1.8362    1.9387
E>> l = l - l_0
ans =
Columns 1 through 10
-0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585
Columns 11 through 19
-0.30585    -0.30585    -0.30585    0.30585    -0.42474    -0.30585    -0.30585    -0.30585    -0.30585

```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

was point 4 2 4 7 4, Ok and of course the program has worked on it and got the whole thing right. So you see the whole thing got replaced by point 3 0 5 8 5 except for this guy

(Refer Slide Time: 07:57)

The screenshot shows the MATLAB environment. The workspace contains several variables: `l` (0.30595), `l_0` (1.6933), `l_1` (1.2053), `l_2` (1.8362), `l_3` (1.9387), `l_4` (1.9387), `l_5` (1.9387), `l_6` (1.9387), `l_7` (1.9387), `l_8` (1.9387), `l_9` (1.9387), `l_10` (1.9387), `l_11` (1.9387), `l_12` (1.9387), `l_13` (1.9387), `l_14` (1.9387), `l_15` (1.9387), `l_16` (1.9387), `l_17` (1.9387), `l_18` (1.9387), `l_19` (1.9387). The command window shows the following operations:

```

E>> l = l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + l_8 + l_9 + l_10 + l_11 + l_12 + l_13 + l_14 + l_15 + l_16 + l_17 + l_18 + l_19
ans =
Columns 1 through 10
0.42474    0.48702    1.2288    2.2786    1.4251    0.4529    2.0249    0.89024    1.9159    1.2021
Columns 11 through 19
1.0822    0.92392    0.56433    -0.44956    0.30585    1.4937    1.2053    1.8362    1.9387
E>> l = l - l_0
ans =
Columns 1 through 10
-0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585    -0.30585
Columns 11 through 19
-0.30585    -0.30585    -0.30585    0.30585    -0.42474    -0.30585    -0.30585    -0.30585    -0.30585

```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

which is positive sign, everything else is

(Refer Slide Time: 07:59)

The screenshot shows the MATLAB IDE with a workspace containing several variables: `abs`, `abs1`, `abs2`, `abs3`, `abs4`, `abs5`, `abs6`, `abs7`, `abs8`, `abs9`, `abs10`, `abs11`, `abs12`, `abs13`, `abs14`, `abs15`, `abs16`, `abs17`, `abs18`, `abs19`, `abs20`, `abs21`, `abs22`, `abs23`, `abs24`, `abs25`, `abs26`, `abs27`, `abs28`, `abs29`, `abs30`, `abs31`, `abs32`, `abs33`, `abs34`, `abs35`, `abs36`, `abs37`, `abs38`, `abs39`, `abs40`, `abs41`, `abs42`, `abs43`, `abs44`, `abs45`, `abs46`, `abs47`, `abs48`, `abs49`, `abs50`, `abs51`, `abs52`, `abs53`, `abs54`, `abs55`, `abs56`, `abs57`, `abs58`, `abs59`, `abs60`, `abs61`, `abs62`, `abs63`, `abs64`, `abs65`, `abs66`, `abs67`, `abs68`, `abs69`, `abs70`, `abs71`, `abs72`, `abs73`, `abs74`, `abs75`, `abs76`, `abs77`, `abs78`, `abs79`, `abs80`, `abs81`, `abs82`, `abs83`, `abs84`, `abs85`, `abs86`, `abs87`, `abs88`, `abs89`, `abs90`, `abs91`, `abs92`, `abs93`, `abs94`, `abs95`, `abs96`, `abs97`, `abs98`, `abs99`, `abs100`. The Command Window shows the following code:

```

E>> treg(1,1)
ans =
Columns 1 through 10
    0.42474    0.49702    1.2288    2.2766    1.4251    0.4329    2.0249    0.89024    1.9159    1.2021
Columns 11 through 19
    1.0822    0.92392    0.56433    -0.44956    0.30585    1.4937    1.2053    1.8362    1.9387
E>> treg(1,1)
ans =
Columns 1 through 10
   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585   -0.30585
Columns 11 through 19
   -0.30585   -0.30585   -0.30585   0.30585   -0.42474   -0.30585   -0.30585   -0.30585   -0.30585
E>>

```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, IIT Madras, with the text "Debugging and Improvements to the MATLAB Implementation".

negative sign and the whole thing is working correctly, Ok

So we are expecting this to work correctly. So you can finish off the minsum loop and

(Refer Slide Time: 08:07)

The screenshot shows the MATLAB IDE with the following code in the Command Window:

```

if B(ij,col) == -1
    ti = ti + 1;
    R1 = R1 + 1;
    %Subtraction
    t(i(i-1)*r+1:col) = t(i(i-1)*r+1:col) - R(i,i);
    %row alignment and store in treg
    treg(ti,1) = min_min(t(i(i-1)*r+1:col),R(i,r));
end
end
end
%Minimum on treg: ti x r
for ii = 1:r
    treg(ii,1) = min(abs(treg(ii,1:11))); %first minimum
    min2 = min(abs(treg(1:rpos-1,pos+1:ti,11))); %second minimum
    p = sign(treg(ii,11));
    parity = prod(p);
    treg(ii,1) = min2; %absolute value for all
    treg(pos,1) = min2; %absolute value for min position
    treg(ii,1) = parity * treg(ii,1); %assign sign
end
%column alignment, addition and store in R
R1 = R1 + ti; %reset the storage counter
ti = 0;
for col = 1:r
    if B(ij,col) == -1
        R1 = R1 + 1;
        ti = ti + 1;
    %column alignment

```

Below the screenshot is a video inset of Prof. Andrew Thangaraj, IIT Madras, with the text "Debugging and Improvements to the MATLAB Implementation".

come out here,



(Refer Slide Time: 08:14)

The image shows a MATLAB window with a matrix displayed in the Command Window. The matrix has 10 columns and 14 rows. The values are floating-point numbers, mostly between -0.30585 and 0.02943. The speaker, Prof. Andrew Thangaraj, is visible in the bottom left corner of the slide.

every column will have only two different values, the minimal value and the next minimal value and the signs are depending on the overall parity, Ok.

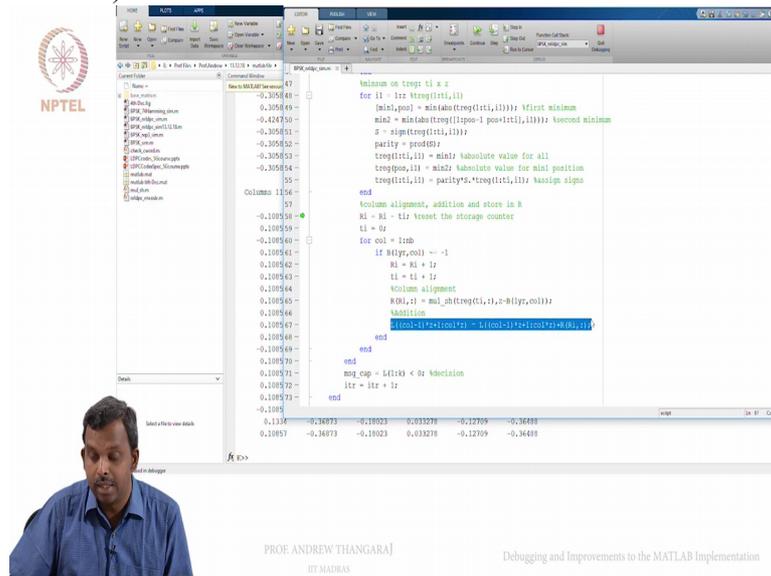
So this is how the minsum has worked. After minsum works, one needs to do the storage back

(Refer Slide Time: 08:29)

The image shows a MATLAB window with code being executed. The code is a loop that iterates over columns of a matrix. It finds the minimum value and its position, then updates the matrix. The speaker, Prof. Andrew Thangaraj, is visible in the bottom left corner of the slide.

into the, into the R array and this is what this loop does for you. And then, L also gets updated,

(Refer Slide Time: 08:38)

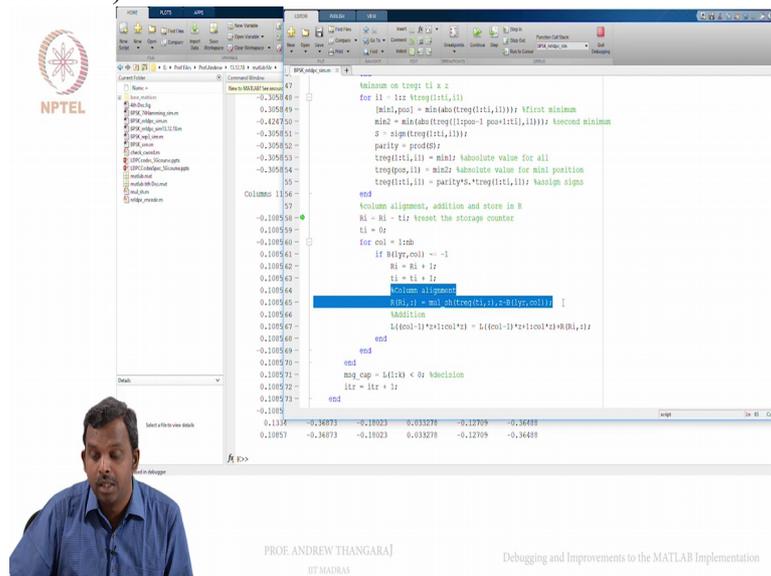


PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok. It gets added and then the storage. For the storage you will do a column alignment.

(Refer Slide Time: 08:41)



PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

You have to multiply by z minus B so that thing aligns up properly, Ok. So this is something you can do and this will also work.

You can, you can try it out and then you will get message cap. So if you run to the cursor, I finish my first

(Refer Slide Time: 08:56)

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

block completely. I have done 10 iterations and you can see message cap. So this message cap, 1 is the question,

(Refer Slide Time: 09:05)

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

it is all zero; you can go through and check it, Ok so it is all zero,



(Refer Slide Time: 09:46)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

0, 0, 0. So now how do you interpret this? Remember what did we type in here? What did we display here?

The display is E b over N naught first,

(Refer Slide Time: 09:55)

NPTEL

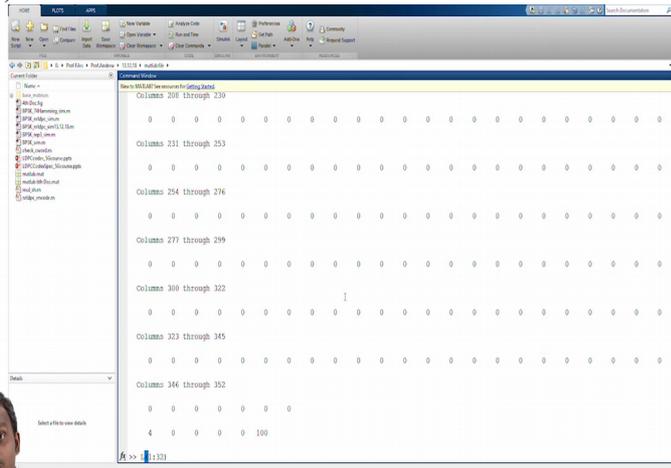
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

which is 4, and then F E R simulated which is 0, B E R simulated is also 0, number of block errors is 0, number of bit errors is 0. So out of these 100 blocks we simulated there were no errors in the output, Ok.

So all the errors were corrected by this. It is also instructive to see this L value. This, remember is the total belief

(Refer Slide Time: 10:16)



The screenshot shows the MATLAB workspace with a large matrix of zeros. The matrix is organized into blocks of columns: Columns 208 through 230, Columns 231 through 253, Columns 254 through 276, Columns 277 through 299, Columns 300 through 322, Columns 323 through 345, and Columns 346 through 352. The last row of the matrix shows a value of 100 in the 346th column. The NPTEL logo is visible in the top left corner.

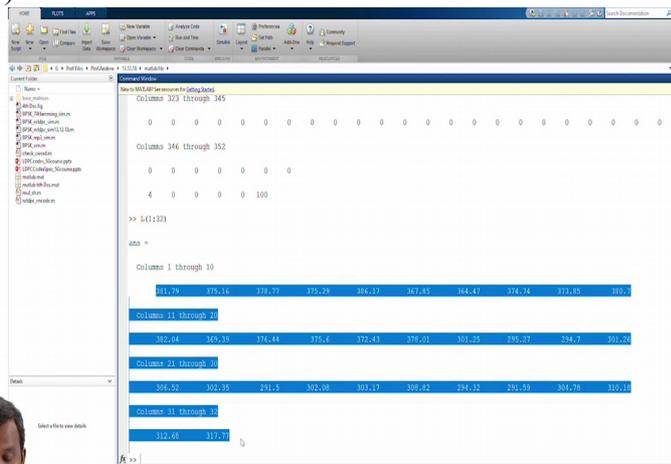


PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

at the end of iterations, 10 iterations. You can see it is all huge, 300 odd and it is all positive, Ok.

(Refer Slide Time: 10:23)



The screenshot shows the MATLAB workspace with a matrix of positive values. The matrix is organized into blocks of columns: Columns 323 through 345, Columns 346 through 352, and Columns 1 through 10. The last row of the matrix shows a value of 100 in the 346th column. The NPTEL logo is visible in the top left corner.



PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

So at the decoder everything has worked and it has also worked really very well.

And when you say it is one you are confident that the bit is 0, there is no doubt. Belief has really increased to a large value at the end of the iteration.

So maybe we should see this a little bit more. May be, maybe we will see for a lower E b over N naught, Ok. So I will keep E b over N naught as 0

(Refer Slide Time: 10:42)

```
SPK_Matlab.m (1)
1  SNRdB = 0;
2  Matices = 0;
Columns 3:
3  Matices = 0;
4  load base_matrices/MR_1_16.txt
5  B = MR_1_16;
6  [Mb, Nb] = size(B);
Columns 6:
7  z = 16;
8
9  SNr = sum(B(i)~=1); Number of non-1 in B
10  lmsg = zeros(max(sum(B == -1,2)),1); %register storage for message
Columns 11:
11
12  k = (Mb-Nb)*z; Number of message bits
13  n = Nb*z; Number of codeword bits
14
15  Rate = k/n;
Columns 16:
16  RSNR = 10^(SNRdB/10);
17  sigma = sqrt(1/(2*Rate*RSNR));
381,7,18
19  Mbitters = 0; Nbitters = 0; Nlocks = 100;
Columns 11:20
20  for i = 1: Nlocks
21  msg = randi(0:1,1,k); %generate random k-bit message
22  msg = zeros(1,k); %all-zero message
23  %encoding
Columns 21:24
24  cword = zeros(1,n); %all-zero codeword
25
26  s = 1 - 2 * cword; %BPSK bit to symbol conversion
27  z = s + sigma * randn(1,n); %AWGN channel I
Columns 31:
312,60  317,77
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

and instead of keeping the

(Refer Slide Time: 10:45)

```
SPK_Matlab.m (1)
1  SNRdB = 0;
2  Matices = 0;
Columns 3:
3  Matices = 0;
4  load base_matrices/MR_1_16.txt
5  B = MR_1_16;
6  [Mb, Nb] = size(B);
Columns 6:
7  z = 16;
8
9  SNr = sum(B(i)~=1); Number of
10  lmsg = zeros(max(sum(B == -1,2)),1); %register storage for message
Columns 11:
11
12  k = (Mb-Nb)*z; Number of message bits
13  n = Nb*z; Number of codeword bits
14
15  Rate = k/n;
Columns 16:
16  RSNR = 10^(SNRdB/10);
17  sigma = sqrt(1/(2*Rate*RSNR));
381,7,18
19  Mbitters = 0; Nbitters = 0; Nlocks = 100;
Columns 11:20
20  for i = 1: Nlocks
21  msg = randi(0:1,1,k); %generate random k-bit message
22  msg = zeros(1,k); %all-zero message
23  %encoding
Columns 21:24
24  cword = zeros(1,n); %all-zero codeword
25
26  s = 1 - 2 * cword; %BPSK bit to symbol conversion
27  z = s + sigma * randn(1,n); %AWGN channel I
Columns 31:
312,60  317,77
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

breakpoint here, I will put the breakpoint here, Ok. So at this point we will put a breakpoint

(Refer Slide Time: 10:52)

PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

And I want to watch this L of 1 colon 32,

(Refer Slide Time: 11:00)

PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok, remember this is after the first iteration. First iteration is already over and you have these kind of beliefs. And remember this is  $0 \text{ dB}$ .  $0 \text{ dB}$   $E_b$  over  $N$  naught, this

(Refer Slide Time: 11:10)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

is sigma of 1 point 2 4 3 2, Ok so it is a huge noise.

Ok the noise has been added a lot and if you see R, R of 1 colon 16 or 32, if you will

(Refer Slide Time: 11:21)

NPTEL

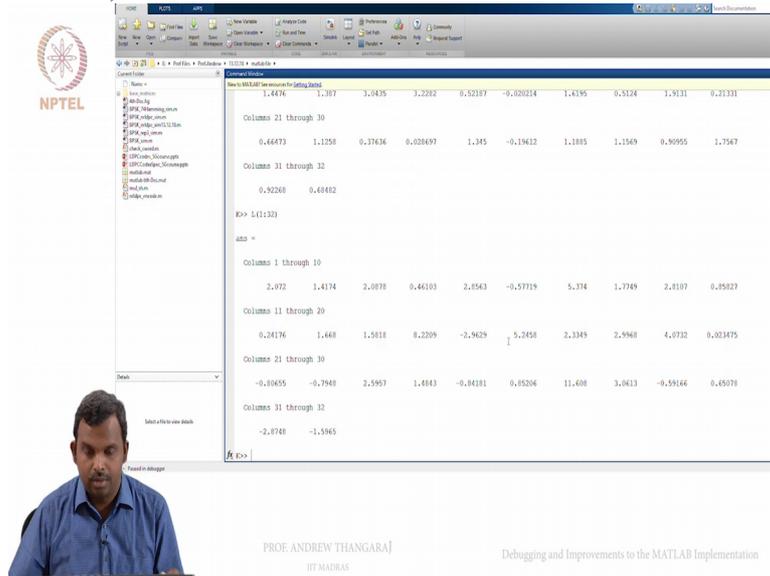
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

that is the received value. Received value, it is quite big values and it will have big errors also.

And if you see L

(Refer Slide Time: 11:30)



The screenshot shows the MATLAB environment. On the left, the workspace contains variables: `l`, `l_err`, `l_err_max`, `l_err_min`, `l_err_std`, `l_err_sum`, `l_err_var`, `l_err_max`, `l_err_min`, `l_err_std`, `l_err_sum`, `l_err_var`, `l_err_max`, `l_err_min`, `l_err_std`, `l_err_sum`, `l_err_var`. The command window displays the following matrix output:

```

Columns 1 through 30
1.4476    1.3977    1.0435    3.2202    0.52197   -0.020214    1.4195    0.5124    1.9131    0.21331

Columns 21 through 30
0.46473    1.1250    0.37436    0.028697    1.345    -0.19412    1.1895    1.1569    0.90955    1.7547

Columns 31 through 32
0.92260    0.68452

ED> l(1:133)

ans =

Columns 1 through 10
2.072    1.4174    2.0878    0.46103    2.0563    -0.57719    5.374    1.7749    2.0107    0.85027

Columns 11 through 20
0.24176    1.440    1.5818    0.2209    -2.9629    5.2458    2.3349    2.4960    4.0732    0.033475

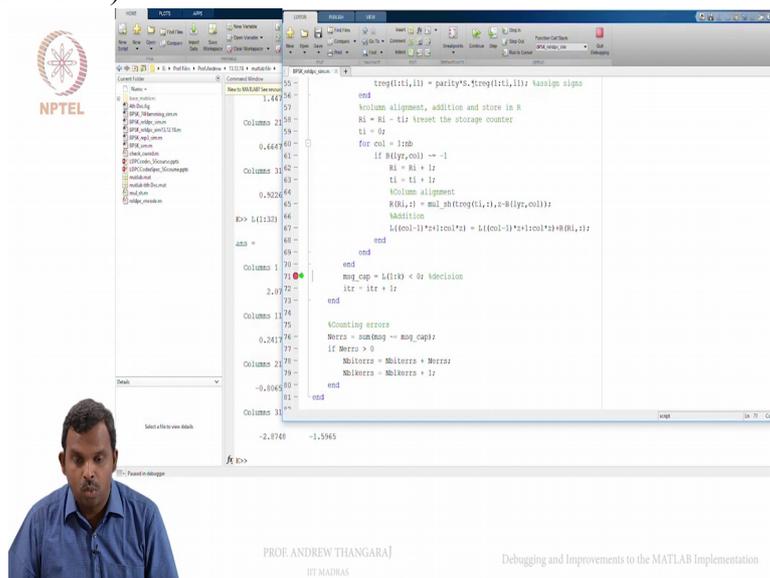
Columns 21 through 30
-0.10455    -0.7940    2.5957    1.4843    -0.84181    0.85204    11.608    3.0413    -0.59146    0.45078

Columns 31 through 32
-2.9740    -1.5945
    
```

PROF ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

that is what happens at the end of the first iteration. So within the first iteration, it looks like this. There are still errors,

(Refer Slide Time: 11:37)



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

```

treg(tl,il) = parity*S_treg(tl,il); %assign signs
end
%column alignment, addition and store in R
Rl = Rl - tl; %reset the storage counter
tl = 0;
for col = 1:nb
    if Rl(:,col) == -1
        Rl = Rl + 1;
        tl = tl + 1;
    end
    %column alignment
    W(Rl,tl) = mul_shftreg(tl,1,2-Rl(:,col));
    %addition
    L((col-1)*s1col+1) = L((col-1)*s1col+1)+Rl(:,tl);
end
end
msg_cap = L(1:K) < 0; %decision
itr = itr + 1;
end
%counting errors
W_errs = sum(msg == msg_cap);
if W_errs > 0
    Nbit_errs = Nbit_errs + W_errs;
    Nbit_errs = Nbit_errs + 1;
end
end
    
```

The workspace contains variables: `l`, `l_err`, `l_err_max`, `l_err_min`, `l_err_std`, `l_err_sum`, `l_err_var`, `l_err_max`, `l_err_min`, `l_err_std`, `l_err_sum`, `l_err_var`. The command window displays the following matrix output:

```

Columns 1 through 10
2.072    1.4174    2.0878    0.46103    2.0563    -0.57719    5.374    1.7749    2.0107    0.85027

Columns 11 through 20
0.24176    1.440    1.5818    0.2209    -2.9629    5.2458    2.3349    2.4960    4.0732    0.033475

Columns 21 through 30
-0.10455    -0.7940    2.5957    1.4843    -0.84181    0.85204    11.608    3.0413    -0.59146    0.45078

Columns 31 through 32
-2.9740    -1.5945
    
```

PROF ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

Ok. Now we can continue this and you get the second iteration. You can see i t r

(Refer Slide Time: 11:41)

```
Columns 31 through 32
0.92268    0.60482

itr =
1

ans =
Columns 1 through 10
2.072    1.4174    2.0078    0.46103    2.8543    -0.57719    5.374    1.7749    2.8107    0.85827

Columns 11 through 20
0.24176    1.648    1.5818    8.2209    -2.9629    5.2458    2.3349    2.9948    4.0732    0.023475

Columns 21 through 30
-0.89455    -0.7948    2.5957    1.4843    -0.84181    0.85204    11.498    3.9613    -0.59144    0.45078

Columns 31 through 32
-2.8748    -1.5945

itr =
1
```

is 1; it has not been incremented yet.

The second iteration you can see

(Refer Slide Time: 11:45)

```
Columns 31 through 32
-2.8748    -1.5945

itr =
1

ans =
Columns 1 through 10
3.8116    13.104    0.89488    0.40347    2.7114    4.3904    7.4349    4.7708    1.7237    3.8138

Columns 11 through 20
-2.2316    10.086    -3.9812    2.5478    -4.1407    2.4023    11.713    4.9741    3.8992    -4.219

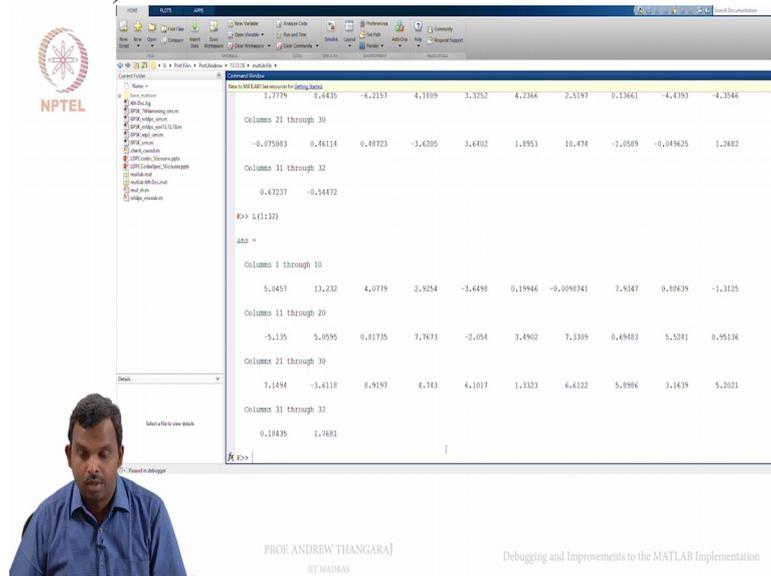
Columns 21 through 30
4.4574    -4.3235    4.3552    2.5309    8.6774    4.9559    1.9292    3.1352    5.1085    4.5787

Columns 31 through 32
-4.8716    -4.1863
```

L, Ok. There are more negative values. Things are not looking very good. Maybe this minus 4, minus 4 and all that, looking a little bit scary.

Let us continue again. Maybe 0 dB is too much noise, Ok. Let us see. We are doing 10 iterations. We never know what can happen. Let us continue again.

(Refer Slide Time: 12:06)



The screenshot shows the MATLAB IDE interface. On the left, the 'Workspace' pane lists variables: 'ans', 'IPSP', 'IPSP\_delay', 'IPSP\_delay\_10', 'IPSP\_delay\_20', 'IPSP\_delay\_30', 'IPSP\_delay\_40', 'IPSP\_delay\_50', 'IPSP\_delay\_60', 'IPSP\_delay\_70', 'IPSP\_delay\_80', 'IPSP\_delay\_90', 'IPSP\_delay\_100', 'IPSP\_delay\_110', 'IPSP\_delay\_120', 'IPSP\_delay\_130', 'IPSP\_delay\_140', 'IPSP\_delay\_150', 'IPSP\_delay\_160', 'IPSP\_delay\_170', 'IPSP\_delay\_180', 'IPSP\_delay\_190', 'IPSP\_delay\_200'. The 'Command Window' displays the following output:

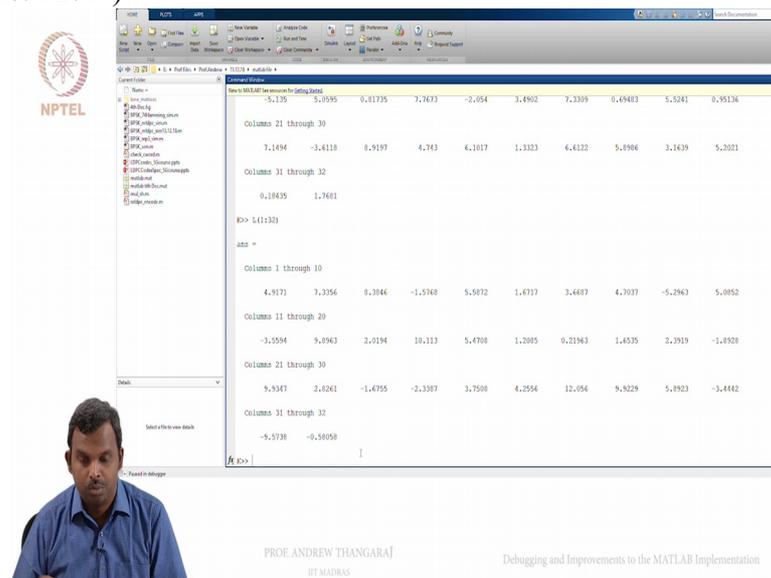
```
ans =  
Columns 1 through 10  
5.0457 13.232 4.0779 2.9254 -3.6498 0.19946 -0.0090741 7.8347 0.08639 -1.3125  
Columns 11 through 20  
-5.135 5.0595 0.81735 7.7673 -2.054 3.4902 7.3309 0.49403 5.5281 0.95136  
Columns 21 through 30  
7.1494 -3.6118 8.9197 4.743 6.1017 1.3323 4.4322 5.8966 3.1439 5.2021  
Columns 31 through 32  
0.18435 1.7681
```

Below the screenshot is a small video inset of Prof. Andrew Thangaraj, a man with a beard wearing a blue shirt, looking at the camera.

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

Maybe, maybe this is some good news. I am not sure. Let us continue again.

(Refer Slide Time: 12:14)



The screenshot shows the MATLAB IDE interface. On the left, the 'Workspace' pane lists variables: 'ans', 'IPSP', 'IPSP\_delay', 'IPSP\_delay\_10', 'IPSP\_delay\_20', 'IPSP\_delay\_30', 'IPSP\_delay\_40', 'IPSP\_delay\_50', 'IPSP\_delay\_60', 'IPSP\_delay\_70', 'IPSP\_delay\_80', 'IPSP\_delay\_90', 'IPSP\_delay\_100', 'IPSP\_delay\_110', 'IPSP\_delay\_120', 'IPSP\_delay\_130', 'IPSP\_delay\_140', 'IPSP\_delay\_150', 'IPSP\_delay\_160', 'IPSP\_delay\_170', 'IPSP\_delay\_180', 'IPSP\_delay\_190', 'IPSP\_delay\_200'. The 'Command Window' displays the following output:

```
ans =  
Columns 1 through 10  
4.9171 7.3356 8.3844 -1.5768 5.0772 1.4717 3.4607 4.7037 -5.2963 5.0052  
Columns 11 through 20  
-3.5594 9.4963 2.0194 10.113 5.4708 1.2085 0.21943 1.4535 2.3919 -1.8928  
Columns 21 through 30  
9.5347 2.8261 -1.6755 -2.3387 3.7508 4.2556 12.056 9.9229 5.8923 -3.4442  
Columns 31 through 32  
-5.5738 -0.58058
```

Below the screenshot is a small video inset of Prof. Andrew Thangaraj, a man with a beard wearing a blue shirt, looking at the camera.

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

Ok, so errors, errors can happen. It can happen that there are some errors that you do not correct. And you go through and see how it is.

So this is the instructive way of seeing

(Refer Slide Time: 12:26)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

what is going on. So you see some of the bits are becoming very

(Refer Slide Time: 12:29)

NPTEL

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

reliable, like 10 and all that. But some bits are still

(Refer Slide Time: 12:32)

NPTEL

PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Column	1	2	3	4	5	6	7	8	9	10
Column 1 through 10	-3.5554	9.4943	2.0194	10.113	5.4708	1.2085	0.21943	1.4535	2.3919	-1.8928
Column 21 through 30	9.9347	2.8261	-1.4755	-2.3387	3.7508	4.2554	12.054	9.9229	5.8923	-3.4442
Column 31 through 32	-9.5730	-0.58058								
Column 1 through 10	10.235	6.8479	8.781	-1.9276	3.103	9.7451	4.484	0.7271	3.7433	
Column 11 through 20	4.2733	11.112	-1.9029	-1.9183	-1.7868	12.022	-2.0987	-4.0323	1.294	-7.8133
Column 21 through 30	3.5917	3.3891	1.0743	3.7027	4.454	3.2265	4.3884	3.4770	-1.3047	1.4227
Column 31 through 32	0.81350	-5.7643								

lying out there, minus, minus these are all errors,

(Refer Slide Time: 12:34)

NPTEL

PROF ANDREW THANGARAJ  
IIT MADRAS

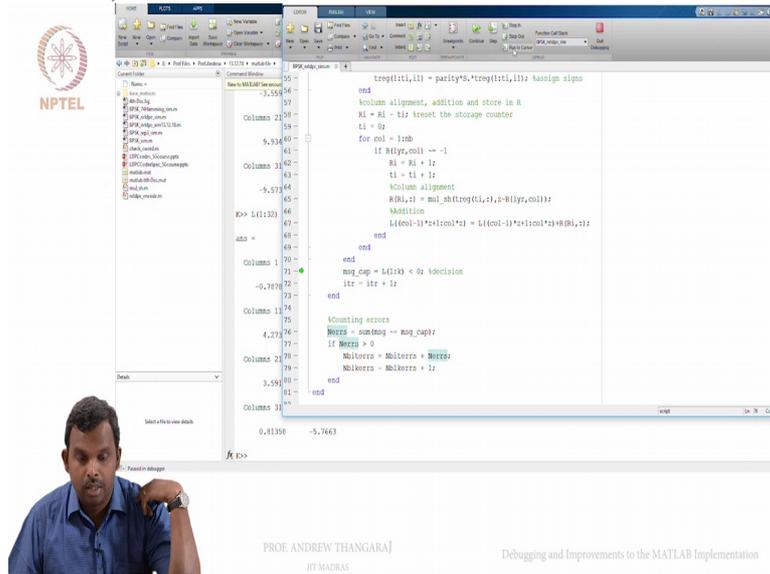
Debugging and Improvements to the MATLAB Implementation

Column	1	2	3	4	5	6	7	8	9	10
Column 1 through 10	-3.5554	9.4943	2.0194	10.113	5.4708	1.2085	0.21943	1.4535	2.3919	-1.8928
Column 21 through 30	9.9347	2.8261	-1.4755	-2.3387	3.7508	4.2554	12.054	9.9229	5.8923	-3.4442
Column 31 through 32	-9.5730	-0.58058								
Column 1 through 10	-0.78781	10.235	6.8479	8.781	-1.9276	3.103	9.7451	4.484	0.7271	3.7433
Column 11 through 20	4.2733	11.112	-1.9029	-1.9183	-1.7868	12.022	-2.0987	-4.0323	1.294	-7.8133
Column 21 through 30	3.5917	3.3891	1.0743	3.7027	4.454	3.2265	4.3884	3.4770	-1.3047	1.4227
Column 31 through 32	0.81350	-5.7643								

Ok. Anything negative is an error, Ok.

So  $0 \leq b \leq N$  ought, for this code, may be at this block length it is not, it is not very good. Let us see that. So if you want to fully continue you can do that Ok here and maybe I will clear this breakpoint

(Refer Slide Time: 12:52)



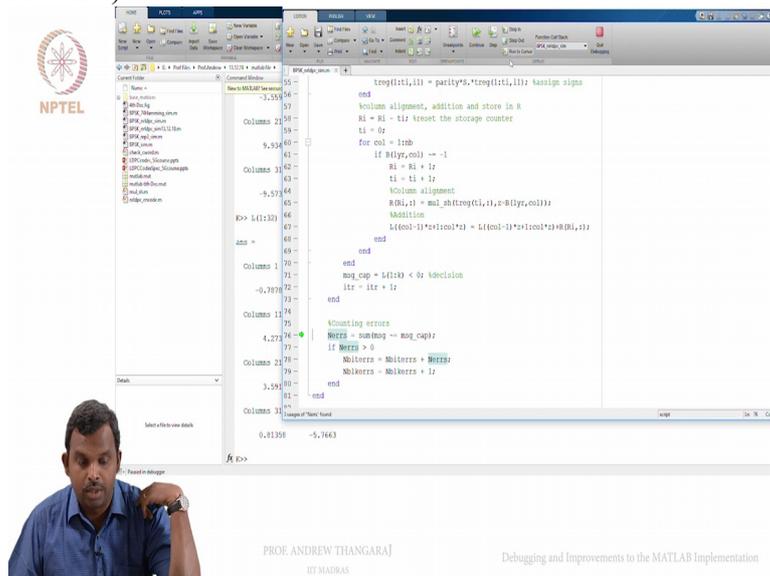
The screenshot shows the MATLAB IDE with the following code visible:

```
55 treq(t(i),l(i)) = parity*S.*treq(t(i),l(i)); %assign signs
56 end
57 %column alignment, addition and store in R
58 R(i) = R(i) - t(i); %reset the storage counter
59 ti = 0;
60 for col = 1:nb
61     if B(i,yr,col) == -1
62         R(i) = R(i) + i;
63         ti = ti + i;
64         %column alignment
65         R(B(i,i)) = mul_sh(treq(t(i),i),z-B(i,yr,col));
66         %addition
67         L((col-1)*r+1:col*i) = L((col-1)*r+1:col*i)+R(B(i,i));
68     end
69 end
70 end
71 msg_cap = L(i:i) < 0; %decision
72 itr = itr + i;
73 end
74
75 %counting errors
76 %errs = sum(msg == msg_cap);
77 if %errs > 0
78     %biterrs = %biterrs + %errs;
79     %biterrs = %biterrs + i;
80 end
81 end
82
83 0.81350 -5.7643
```

Below the code, a man in a blue shirt is shown in a thinking pose. At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ IIT MADRAS Debugging and Improvements to the MATLAB Implementation".

and run to cursor, Ok.

(Refer Slide Time: 12:54)



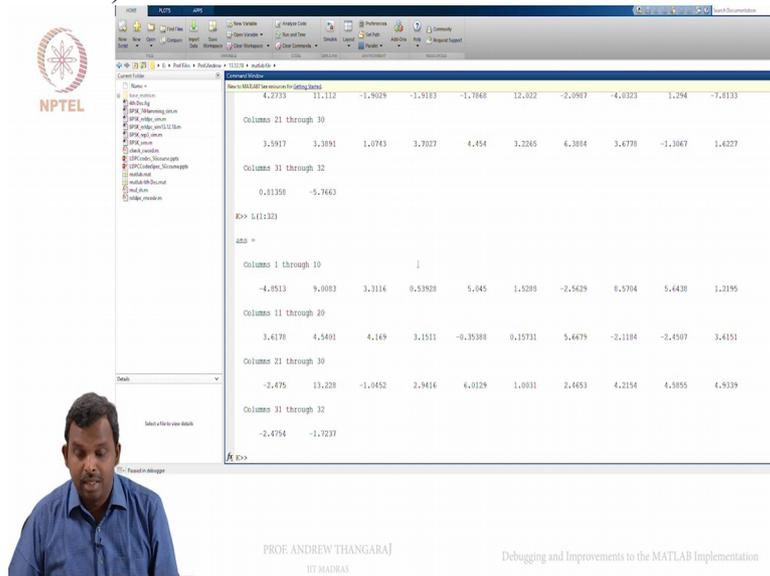
The screenshot shows the MATLAB IDE with the following code visible:

```
55 treq(t(i),l(i)) = parity*S.*treq(t(i),l(i)); %assign signs
56 end
57 %column alignment, addition and store in R
58 R(i) = R(i) - t(i); %reset the storage counter
59 ti = 0;
60 for col = 1:nb
61     if B(i,yr,col) == -1
62         R(i) = R(i) + i;
63         ti = ti + i;
64         %column alignment
65         R(B(i,i)) = mul_sh(treq(t(i),i),z-B(i,yr,col));
66         %addition
67         L((col-1)*r+1:col*i) = L((col-1)*r+1:col*i)+R(B(i,i));
68     end
69 end
70 end
71 msg_cap = L(i:i) < 0; %decision
72 itr = itr + i;
73 end
74
75 %counting errors
76 %errs = sum(msg == msg_cap);
77 if %errs > 0
78     %biterrs = %biterrs + %errs;
79     %biterrs = %biterrs + i;
80 end
81 end
82
83 0.81350 -5.7643
```

Below the code, a man in a blue shirt is shown in a thinking pose. At the bottom of the slide, the text reads: "PROF. ANDREW THANGARAJ IIT MADRAS Debugging and Improvements to the MATLAB Implementation".

So it ran the whole thing and you can see the final update in L.

(Refer Slide Time: 12:59)



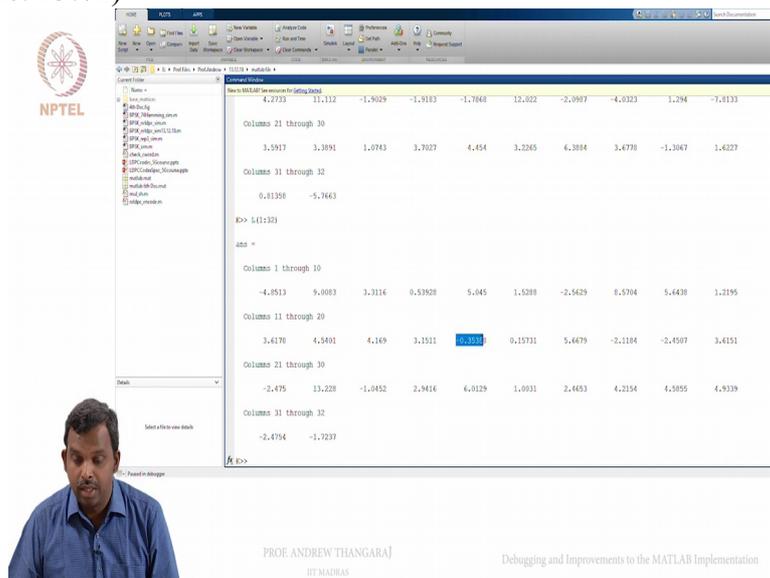
The screenshot shows the MATLAB IDE interface. On the left is the NPTEL logo and a file browser. The main window displays a matrix with the following values:

4.2733	11.1112	-1.9029	-1.9183	-1.7868	12.022	-2.0987	-4.0323	1.294	-7.8133
Columns 21 through 30									
3.5917	3.3891	1.0743	3.7027	4.454	3.2265	6.3884	3.4778	-1.3067	1.4227
Columns 31 through 32									
0.81358	-5.7663								
E>> i(1:133)									
ans =									
Columns 1 through 10									
-4.8513	9.0083	3.3116	0.53928	5.045	1.5289	-2.5629	8.5704	5.6438	1.2195
Columns 11 through 20									
3.4178	4.5401	4.149	3.1511	-0.35388	0.15731	5.4679	-2.1184	-2.4507	3.4151
Columns 21 through 30									
-2.475	13.228	-1.0452	2.9416	6.0129	1.0031	2.4653	4.2154	4.5955	4.9339
Columns 31 through 32									
-2.4754	-1.7237								

At the bottom, the text reads: PROF ANDREW THANGARAJ, IIT MADRAS, Debugging and Improvements to the MATLAB Implementation.

There is still some negatives.

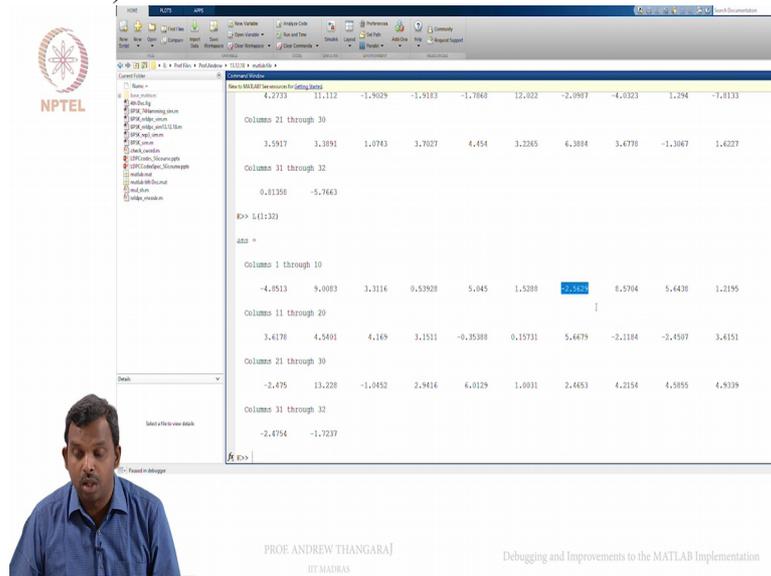
(Refer Slide Time: 13:01)



This screenshot is identical to the previous one, but with a red rectangular box highlighting the value -0.35388 in the matrix. The text at the bottom remains the same: PROF ANDREW THANGARAJ, IIT MADRAS, Debugging and Improvements to the MATLAB Implementation.

Here are still negatives so this will be erroneous block,

(Refer Slide Time: 13:03)



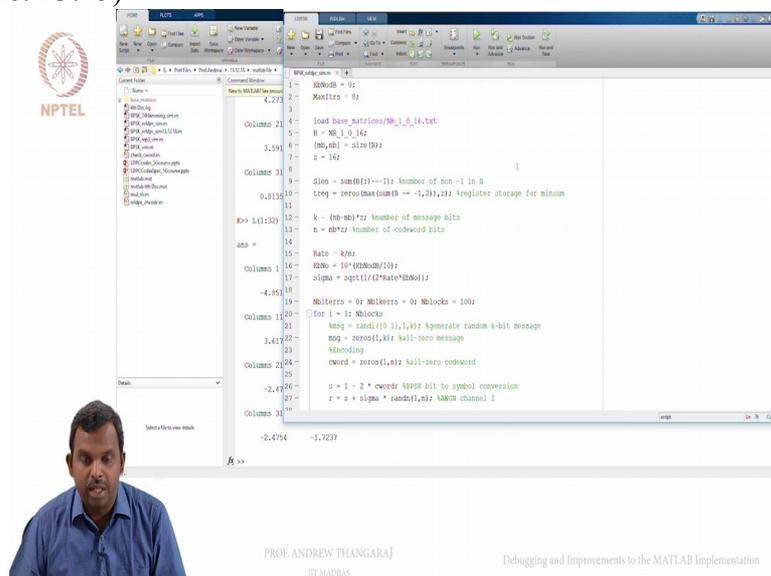
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok.

So, so this is, this is something you can see. So the errors can happen. If you reduce block error rate, errors can happen. And, Ok

(Refer Slide Time: 13:15)



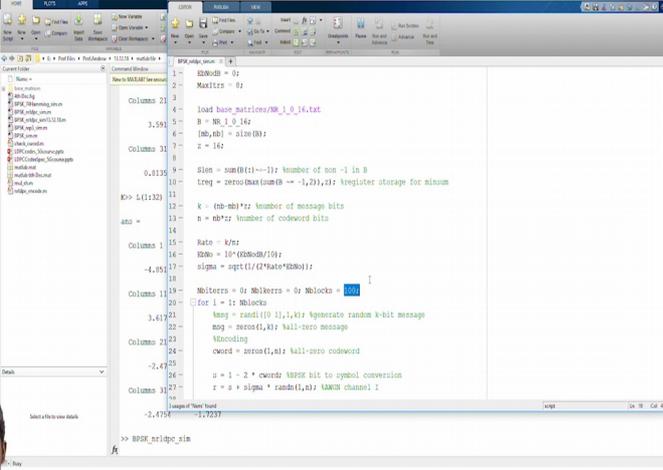
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

I am going to quit debugging here. But maybe we will run this at 0 dB Eb over N naught and see what it does.

It is 8 iterations we did, Ok and for 100 blocks,

(Refer Slide Time: 13:25)

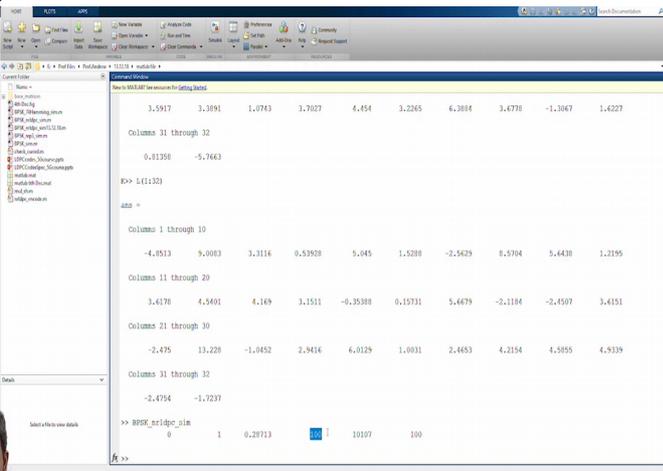



PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

for this Rate point 3 2 something code, block length is not very high. So you see every frame was an error. Ok

(Refer Slide Time: 13:33)

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

so there are 100 block errors but not every bit was an error. Bit error



(Refer Slide Time: 14:01)

NPTEL

```
Columns 31 through 32
0.81358 -5.7663

ED> 141123)
ans =

Columns 1 through 10
-4.9513 5.0083 3.3116 0.53928 5.045 1.5288 -2.5629 8.5704 5.6438 1.2195

Columns 11 through 20
3.6178 4.5401 4.169 3.1511 -0.35388 0.15731 5.6679 -2.1194 -3.4597 3.4151

Columns 21 through 30
-2.475 13.228 -1.0452 2.9416 6.0129 1.0931 2.4653 4.2154 4.5955 4.9339

Columns 31 through 32
-2.4754 -1.7237

>> BERSE_nrl4c_sim
0 1 0.28713 100 10107 100

>> BERSE_nrl4c_sim
1 0.92 0.19849 52 6987 100
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

blocks that got corrected. So it is good news. So 8 iterations

(Refer Slide Time: 14:05)

NPTEL

```
Columns 31
0.81358 load base_matrices/NR_1_16.mat
B = NR_1_16;
ED> 141123)
[Mb,Nb] = size(B);
z = 16;
ans =
8
9 Slen = sum(B(i==1)) Number of non-1 in B
lreq = zeros(max(Slen,1),z); register storage for minimum
Columns 1
-4.9513 k = (Mb-Nb)*z Number of message bits
Columns 11
-2.4720 n = Mb*z Number of codeded bits
Columns 114
3.6178 Rate = k/N;
Columns 21
-2.4720 EbNo = 10*(EbNoBf/10);
Columns 31
-2.4724 sigma = sqrt(1/(2*Rate*EbNo));
Columns 31
-2.4724 MbIters = 0; MbIters = 0; MbLocks = 100;
Columns 31
-2.4720 for l = 1: MbLocks
Columns 31
-2.4720 msg = randi(0:1,1,1); generate random k-bit message
Columns 31
-2.4720 msg = zeros(1,k); fill-zero message
Columns 31
-2.4724 %encoding
Columns 31
-2.4724 cword = zeros(1,n); fill-zero codeword
Columns 31
-2.4720 z = 1 - 2 * cword; BERSE bit to symbol conversion
Columns 31
-2.4720 z = z + sigma * randn(1,n); AWGN channel I
Columns 31
-2.4720
```

PROF. ANDREW THANGARAJ  
IIT MADRAS

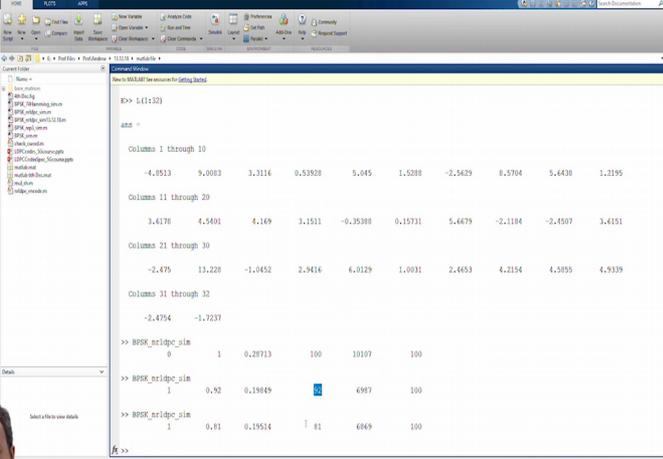
Debugging and Improvements to the MATLAB Implementation

is good.

So one more thing worth trying is maybe you want to try 16 iterations, right. So it will take a little longer. But it is worth seeing. This also tells you that there is lot of scope for improving the efficiency. For instance this loop is very, very



(Refer Slide Time: 14:37)



```
ED> 1(1:100)
ans =
Columns 1 through 10
-4.9513  5.0083  3.3116  0.53928  5.045  1.5288  -2.5629  8.5794  5.6438  1.2195
Columns 11 through 20
3.4178  4.5401  4.149  3.1511  -0.35388  0.15731  5.4679  -2.1184  -2.4507  3.4151
Columns 21 through 30
-2.475  13.228  -1.0452  2.9416  6.0129  1.0031  2.4653  4.2154  4.5855  4.9339
Columns 31 through 32
-2.4754  -1.7237
>> BPSK_errldc_slm
    0  0.28713  100  10107  100
>> BPSK_errldc_slm
    1  0.92  0.19849  6987  100
>> BPSK_errldc_slm
    1  0.81  0.19514  6869  100
```



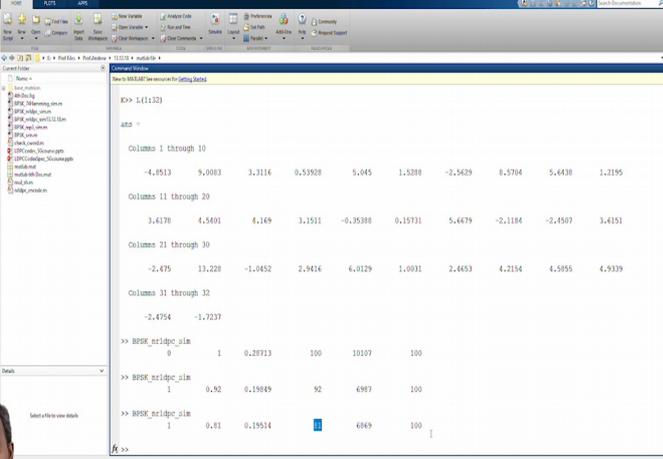
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

improvement, Ok.

Remember this is not the same 100 blocks. This is another 100 blocks but out of 92 errors, 92 out of 100 were in error here.

(Refer Slide Time: 14:45)



```
ED> 1(1:100)
ans =
Columns 1 through 10
-4.9513  5.0083  3.3116  0.53928  5.045  1.5288  -2.5629  8.5794  5.6438  1.2195
Columns 11 through 20
3.4178  4.5401  4.149  3.1511  -0.35388  0.15731  5.4679  -2.1184  -2.4507  3.4151
Columns 21 through 30
-2.475  13.228  -1.0452  2.9416  6.0129  1.0031  2.4653  4.2154  4.5855  4.9339
Columns 31 through 32
-2.4754  -1.7237
>> BPSK_errldc_slm
    0  0.28713  100  10107  100
>> BPSK_errldc_slm
    1  0.92  0.19849  92  6987  100
>> BPSK_errldc_slm
    1  0.81  0.19514  6869  100
```



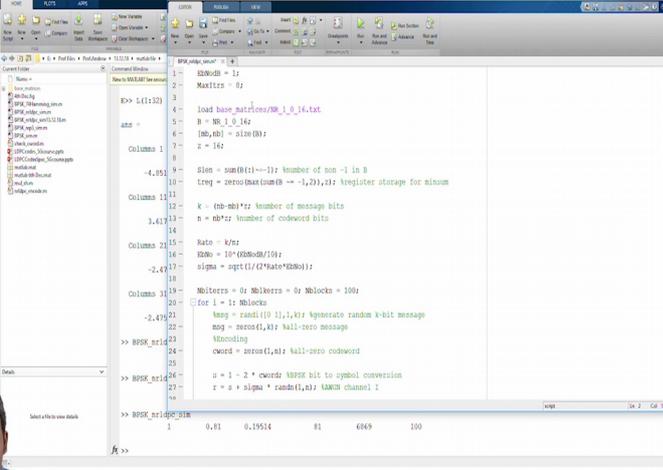
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Here 81 out of 100 were in error. Fraction wise may be not a big difference, point 8, point 9 is pretty much the same at this kind of thing but there is improvement.

So there is merit in doing some more iterations, so but we will keep the number of iterations at 8. I do not want to do more than 8.

(Refer Slide Time: 15:00)

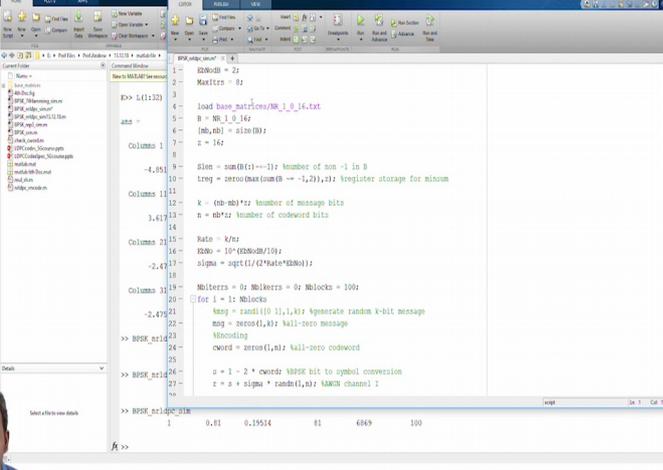


PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

May be we will increase the  $E_b$  over  $N$  naught to 2.

(Refer Slide Time: 15:03)



PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

And let us see what happens. Ok

So in the meantime let us think about how this, how this loop can be speeded up, Ok.

(Refer Slide Time: 15:11)

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

So I am running it through 1 to n b and I am running this loop only for those columns which have, which are not minus 1. So, so very easy way to speed this up is to do the following. So you write col to be equal to, Ok not 1 colon n b, those values of b of layer comma col which are not minus 1, Ok.

So one very easy way to do this is to write this over this thing, Ok. Find of not minus 1

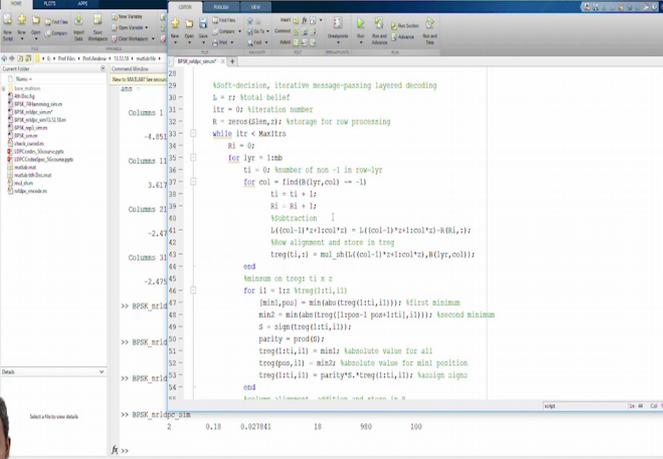
(Refer Slide Time: 15:43)

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

and then you got rid of one loop,

(Refer Slide Time: 15:46)



PROF. ANDREW THANGARAJ  
IIT MADRAS

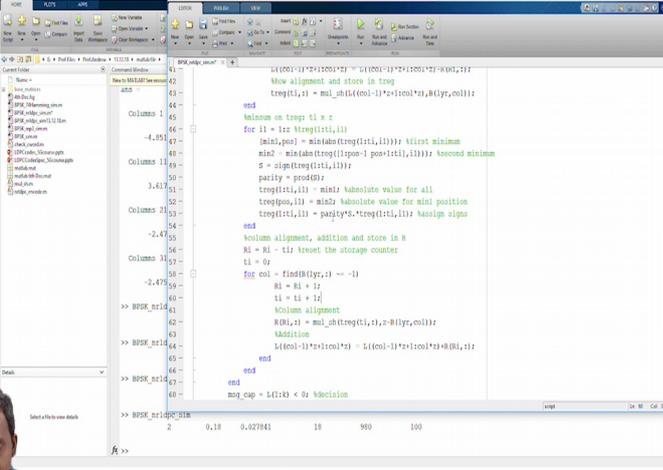
Debugging and Improvements to the MATLAB Implementation

Ok.

So this, what does this do here? So it looks at B of layer comma col, no sorry colon, you should change this to colon. It looks at the whole row, the layer block row and see where all there is no minus 1 and simply loops over that, Ok.

So this is a very quick way to cut short this little loop here. I can do the same thing here, comma colon and then I

(Refer Slide Time: 16:16)

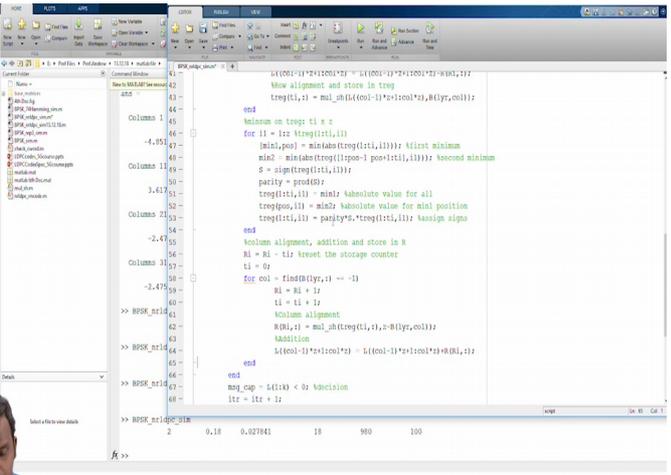


PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

get rid of this. So this should be a little bit faster,

(Refer Slide Time: 16:19)



The screenshot shows a MATLAB script with the following key sections:

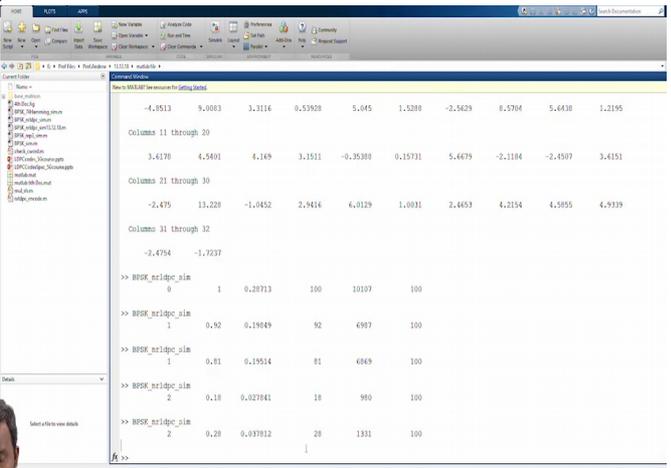
```
l1(001-1)*i+1(001)*j = l1(001-1)*i+1(001)*j+8(8),1;1;
Show alignment and store in treg
treg(tl,1) = min_shft((col-1)*i+1(001)*j,8(8),col);
end
Minimum on treg: ti x z
for il = 1:r2(treg(tl,1))
    [min1,pos] = minabs(treg(tl,1)); %first minimum
    min2 = minabs(treg(tl,1-pos+1(001)*j)); %second minimum
    S = sign(treg(tl,1));
    parity = prod(S);
    treg(tl,il) = min1; %absolute value for all
    treg(pos,il) = min2; %absolute value for min1 position
    treg(tl,il) = parity*S.*treg(tl,il); %assign sign
end
%column alignment, addition and store in B
Rl = Rl - ti; %reset the storage counter
tl = 0;
for col = find(B(lyr,1) == -1)
    Rl = Rl + 1;
    tl = tl + 1;
end
%column alignment
R(Rl,1) = min_shft(treg(tl,1),2-8(8),col);
%addition
l1((col-1)*i+1(001)*j) = l1((col-1)*i+1(001)*j+8(8),1);
end
end
msg_cap = l1(k) < 0; %decision
itr = itr + 1;
end
end
```

Below the code, the speaker's name and affiliation are listed: PROE ANDREW THANGARAJ, IIT MADRAS. The slide title is "Debugging and Improvements to the MATLAB Implementation".

Ok. And hopefully I did not make any mistakes. So let us run it again and check this.

Ok so

(Refer Slide Time: 16:37)



The screenshot shows the MATLAB command window with the following output:

```
Columns 11 through 20
-4.1513    5.0083    1.3116    0.53028    5.045    1.5288    -2.5629    0.5706    5.6418    1.2195

Columns 21 through 30
3.4170    4.5401    4.169    3.1511    -0.35388    0.15731    5.4679    -2.1184    -2.4507    3.4151

Columns 31 through 32
-2.475    13.228    -1.0452    2.9416    6.0129    1.0031    2.4653    4.2154    4.9555    4.9339

Columns 31 through 32
-2.4754    -1.7237

>> BPSK_nrdpc_min
0    1    0.28713    100    10107    100

>> BPSK_nrdpc_min
1    0.92    0.19049    92    6907    100

>> BPSK_nrdpc_min
1    0.81    0.19514    81    6869    100

>> BPSK_nrdpc_min
2    0.18    0.027841    18    980    100

>> BPSK_nrdpc_min
2    0.28    0.037812    28    1331    100
```

Below the output, the speaker's name and affiliation are listed: PROE ANDREW THANGARAJ, IIT MADRAS. The slide title is "Debugging and Improvements to the MATLAB Implementation".

there were 38 errors this time, so that is Ok, so that is Ok, 28, 18 is not a big deal. I run it again. Hopefully this works a little bit faster, not sure. Ok. Anyway loops in MATLAB are not always the best thing to do.

So we need to figure out how to do this differently from, without loop. So in fact it is possible to write without all of these loops, Ok. So you can write some very efficient MATLAB code but we are not trying to do that much.

So anyway this is good, this fast enough for us. It is not too bad. We can work with this. So this gives you a 2 d B, it gives you a certain block, a frame error rate already, just point 2 2, Ok

(Refer Slide Time: 17:15)

The screenshot shows the MATLAB environment. On the left is the NPTEL logo. The main window displays the following data:

```

Columns 11 through 20
    3.4170    4.5401    4.169    3.1511   -0.35388    0.15731    5.4679   -2.1184   -2.4507    3.4151

Columns 21 through 30
   -2.475    13.228   -1.0452    2.9416    6.0129    1.0031    2.4653    4.2154    4.5955    4.9339

Columns 31 through 32
   -2.4754   -1.7237

>> BPSK_mriqc_sim
    0      1    0.28713    100    10107    100

>> BPSK_mriqc_sim
    1    0.92    0.19049    92    6987    100

>> BPSK_mriqc_sim
    1    0.81    0.19514    81    6869    100

>> BPSK_mriqc_sim
    2    0.18    0.027841    18    900    100

>> BPSK_mriqc_sim
    2    0.28    0.037812    28    1311    100

>> BPSK_mriqc_sim
    2    0.22    0.037528    22    1321    100
  
```

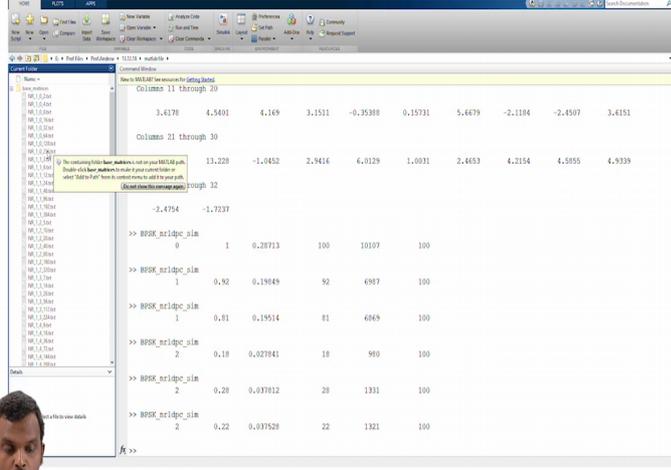
Below the MATLAB window is a video inset of Prof. Andrew Thangaraj, with the text "PROF. ANDREW THANGARAJ, IIT MADRAS" and "Debugging and Improvements to the MATLAB Implementation" below it.

and then slowly it will start falling, 3 d B of E b over N naught and all that, and it will start falling to lower numbers, Ok.

So hopefully this is expected performance. I am not sure if this is really good. I will have to check this. It looks; looks like this code should do something slightly better than this but may be the expansion factor is very low but at least as high that it works. And I will check on this once again and it should be fine, Ok.

So, so maybe we need to change this to some other, some other base matrix. So let us see what all base matrix we have? So if we look at 1 0 it goes all the way up to 256, so let us try

(Refer Slide Time: 17:57)



Columns 11 through 20

3.4178	4.5401	4.1169	3.1511	-0.35388	0.15731	5.4679	-2.1184	-2.4507	3.4151
--------	--------	--------	--------	----------	---------	--------	---------	---------	--------

Columns 21 through 30

13.228	-1.0452	2.9416	6.0129	1.0031	2.4653	4.2154	4.5555	4.9339	
--------	---------	--------	--------	--------	--------	--------	--------	--------	--

Columns 31 through 40

-2.4754	-1.7237								
---------	---------	--	--	--	--	--	--	--	--

Columns 41 through 50

0	1	0.28713	100	10107	100				
---	---	---------	-----	-------	-----	--	--	--	--

Columns 51 through 60

1	0.92	0.19649	92	6987	100				
---	------	---------	----	------	-----	--	--	--	--

Columns 61 through 70

1	0.61	0.19514	81	6869	100				
---	------	---------	----	------	-----	--	--	--	--

Columns 71 through 80

2	0.18	0.027841	18	900	100				
---	------	----------	----	-----	-----	--	--	--	--

Columns 81 through 90

2	0.28	0.037812	28	1331	100				
---	------	----------	----	------	-----	--	--	--	--

Columns 91 through 100

2	0.22	0.037528	22	1321	100				
---	------	----------	----	------	-----	--	--	--	--

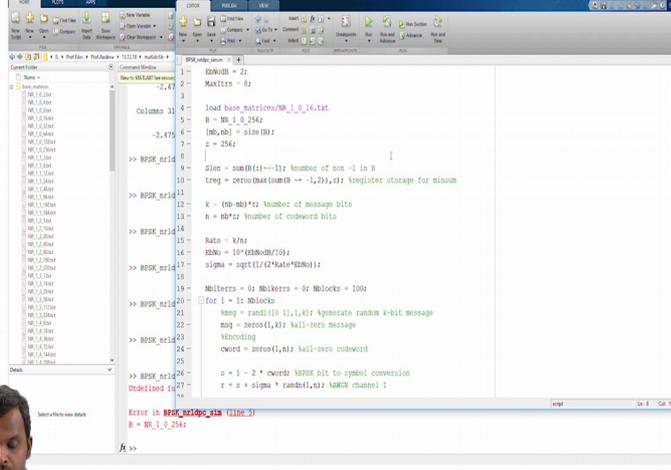
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

1 0 256, Ok. This will take a little longer to run.

Ok. I do not think there need to make any other change here. This should just work. This will be a much larger block length but nevertheless, keeping the base matrix the same so the complexity of this should not be very high, I think so let us run this.

(Refer Slide Time: 18:22)



```
1: Nblocks = 10;  
2: MaxBits = 0;  
3:  
4: load base_matrices/MB_1_0_16.txt  
5: B = MB_1_0_256;  
6: [nb,nb] = size(B);  
7: z = 256;  
8:  
9: |  
10: zlen = sum(B(:) == -1); number of ones -1 in B  
11: treq = zeros(max(zlen, z)); register storage for minsum  
12:  
13: k = (nb-nb)*z; number of message bits  
14: n = nb*z; number of codeword bits  
15:  
16: Rate = k/n;  
17: sigma = sqrt(1/(2*Rate*(256-1)));  
18:  
19: NbitsPerS = 0; Nblocks = 0; Nblocks = 100;  
20:  
21: for i = 1: Nblocks  
22:     msg = randi(0:1,1,1); generate random k-bit message  
23:     msg = zeros(1,k); fill-zero message  
24:     cword = zeros(1,n); fill-zero codeword  
25:  
26:     a = 1 - 2 * cword; BPSK bit to symbol conversion  
27:     z = z + sigma * randn(1,n); AWGN channel |  
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:  

```

(Refer Slide Time: 18:29)

The image is a composite. On the left is the NPTEL logo. In the center is a screenshot of a MATLAB script titled 'LDPC.m'. The script defines parameters for an LDPC code, including  $N=256$ ,  $k=100$ , and  $r=2$ . It includes comments for the number of non-1s in B, the number of message bits, and the number of codeword bits. The script also defines the rate, SNR, and sigma. A loop is shown for  $i=1:Nblocks$  where a random message is generated and converted to a symbol. At the bottom, there is a red error message: 'Error in LDPC\_sim (LINE 3) B = NR\_1\_0\_256;'. Below the screenshot is a portrait of Prof. Andrew Thangaraj, with his name and 'IIT MADRAS' written below it. To the right of the portrait, the text 'Debugging and Improvements to the MATLAB Implementation' is visible.

256 here, Ok. So let us see. This is a much larger block length. And let us see if this works correctly. Here you can see it is taking a little bit of time. So generally LDPC codes at larger block lengths will perform much, much better.

Ok so block length of 1000 is a bit low for LDPC codes, 2000, 3000 all they are very difficult to beat. They are really very, very good. They give very competitive performance.

So 256 is a very huge block length you can imagine. So this will, program will take a little bit longer. But let us see.  $2d \text{ B E b over } N \text{ naught } 256$  does it give you significantly better performance than before, Ok? Let us see.

Sort of expecting zero errors here, let us see. Ok so you can see there was just 1 error here

(Refer Slide Time: 19:33)

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

```
-2.4754 -1.7237
>> BPSK_modc_sim
0 1 0.28713 100 10107 100
>> BPSK_modc_sim
1 0.92 0.19849 92 6987 100
>> BPSK_modc_sim
1 0.81 0.19514 81 6869 100
>> BPSK_modc_sim
2 0.18 0.027841 18 960 100
>> BPSK_modc_sim
2 0.28 0.037812 28 1331 100
>> BPSK_modc_sim
2 0.22 0.03528 22 1321 100
>> BPSK_modc_sim
Undefined function or variable 'm_1_0_254'.
Error in BPSK_modc_sim (line 5)
B = m_1_0_254;
>> BPSK_modc_sim
2 0.01 5.5043e-05 31 100
```

PROF. ANDREW THANGARAJ  
IIT MADRAS  
Debugging and Improvements to the MATLAB Implementation

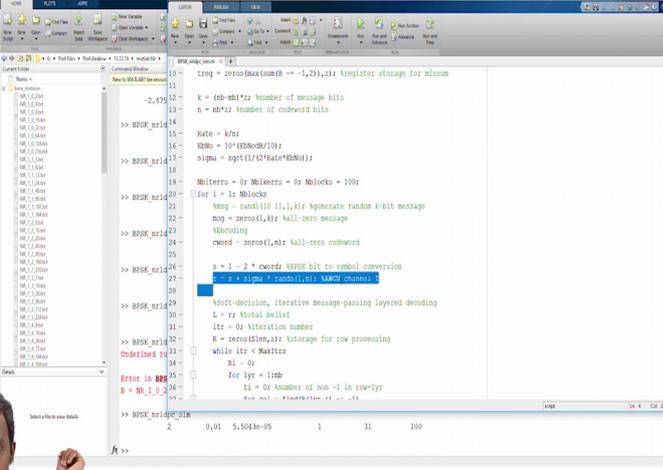
Ok, out of 100, that actually looks quite wrong to me. It should be much, much better than this.

So, so not sure if may be, this is expected performance. So I will look into it little bit more compare with, with some literature and confirm whether this is good or not. But I think this should be good. It looks alright to me as far as the code is concerned. I do not think there is any big mistake, Ok.

The few changes that usually people make to this code. The first change is rate match. So you have to do rate matching. So this always simulates the lowest rate that is there. But that is not how transmission happens, Ok.

So you have to do some rate matching. That is the first thing. The other thing that people do is you want to do; you want to, you know quantize the received value. Ok so at this point you can see,

(Refer Slide Time: 20:26)

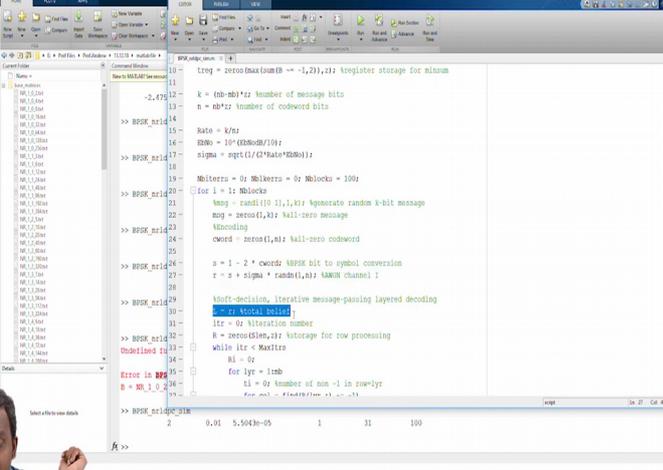


PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

so I am doing A W G N channel, right. So I am using the real value itself as

(Refer Slide Time: 20:32)



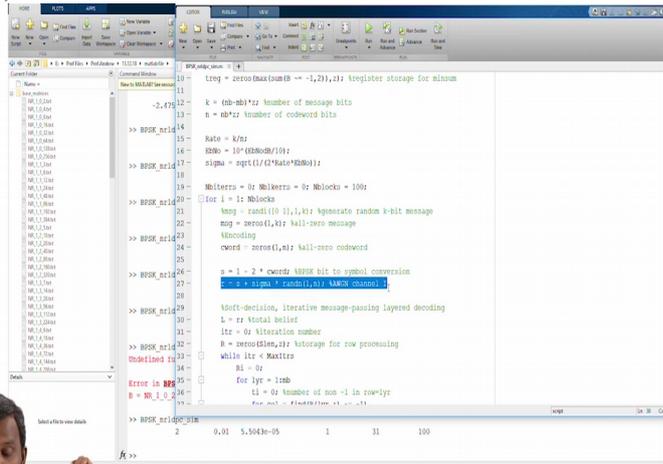
PROF ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

L, Ok. So that is not very nice.

So you want to, you want to quantize

(Refer Slide Time: 20:37)



The screenshot shows MATLAB code for BPSK modulation and iterative decoding. The code includes parameters for message bits (k), codeword bits (n), and modulation order (M). It generates a random message, adds a tail-zero codeword, and performs soft-decision, iterative message-passing layered decoding. The code uses a while loop for iterations and a for loop for layers. The final output is the decoded message.

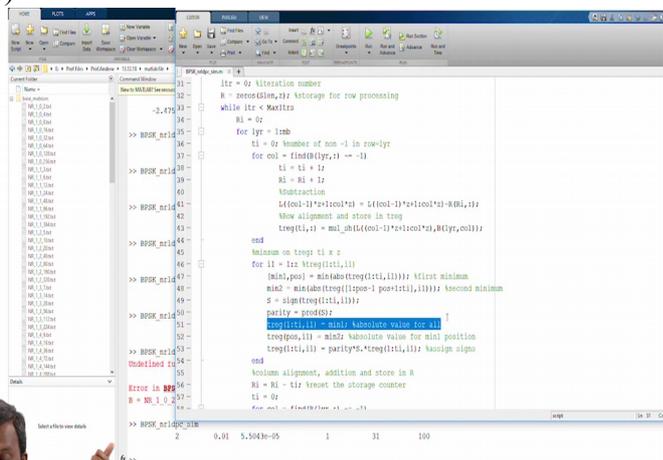
PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

this, Ok to how many error bits or integer value between say minus 256 and 256 if you want to use 8 bit integer values, Ok. So that is something very important. You have to quantize, number 1. Like I said number 1 was rate matching. I will talk about it in the next lecture.

And after that quantization, which will also be there in, hopefully in the next lecture. And the last step is changing this step to,

(Refer Slide Time: 21:02)



The screenshot shows MATLAB code for iterative decoding. The code includes parameters for iteration number (itr), storage for row processing (R), and modulation order (M). It performs a while loop for iterations and a for loop for layers. The code uses a while loop for iterations and a for loop for layers. The final output is the decoded message. A line of code is highlighted: `trej(tl,il) = min(abs(trej(tl,il)));`

PROF. ANDREW THANGARAJ  
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

to offset minsum. Ok, so instead of doing min 1, min 2 you want to subtract a small offset here, Ok.

So all these three we will do in the next lecture and see how much it improves or how the performance changes. Ok so other than that the basic algorithm is as such Ok. Thank you very much.