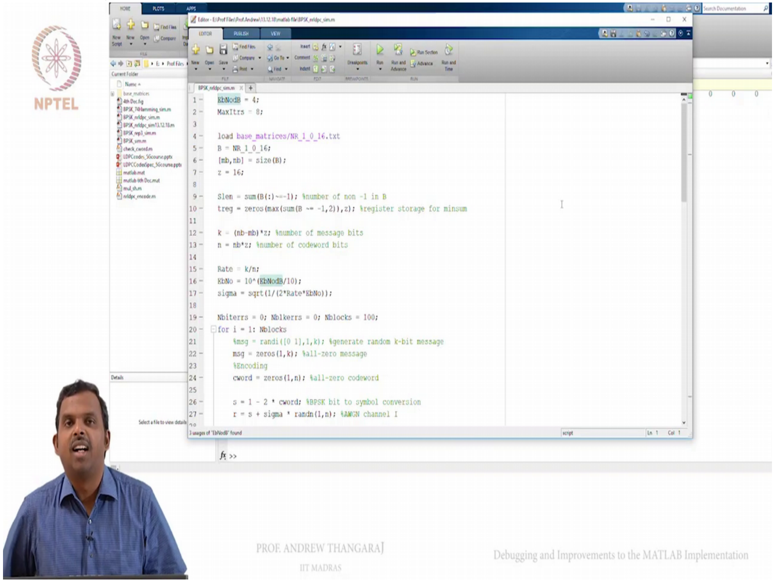
LDPC and Polar codes in 5G Standard
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
Debugging and Improvements to the MATLAB Implementation

(Refer Slide Time: 00:15)


Editor - E:\Prof Files\Prof.Andrew\13.12.18\matlab file\BPSK_nrldpc_sim.m
EDITOR
PUBLISH
VIEW
BPSK_nrldpc_sim.m
NPTEL
PROF. ANDREW THANGARAJ
IIT MADRAS
Debugging and Improvements to the MATLAB Implementation

```
1  EbNodB = 4;
2  MaxItrs = 8;
3
4  load base_matrices/NR_1_0_16.txt
5  B = NR_1_0_16;
6  [mb,nb] = size(B);
7  z = 16;
8
9  Slen = sum(B(:)~=-1); %number of non -1 in B
10 treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum
11
12 k = (nb-mb)*z; %number of message bits
13 n = nb*z; %number of codeword bits
14
15 Rate = k/n;
16 EbNo = 10^(EbNodB/10);
17 sigma = sqrt(1/(2*Rate*EbNo));
18
19 Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
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     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 I
```

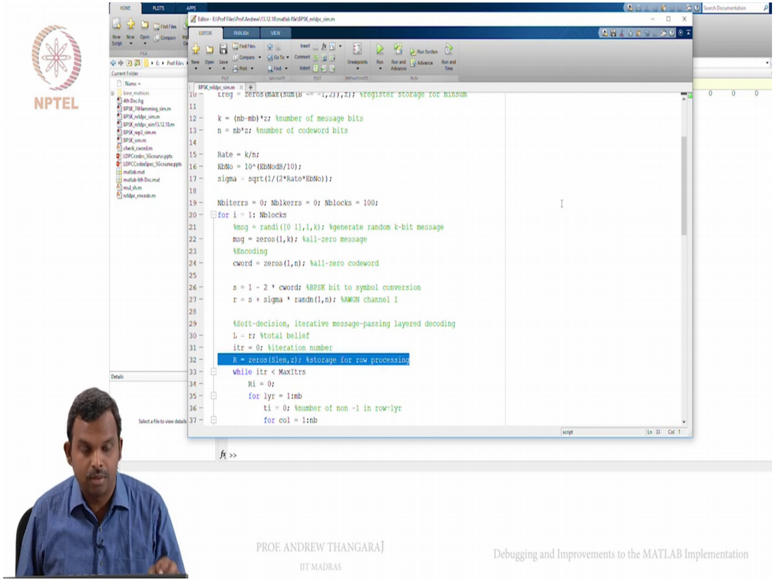
```
3 usages of "EbNodB" found
fx >>
```

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)



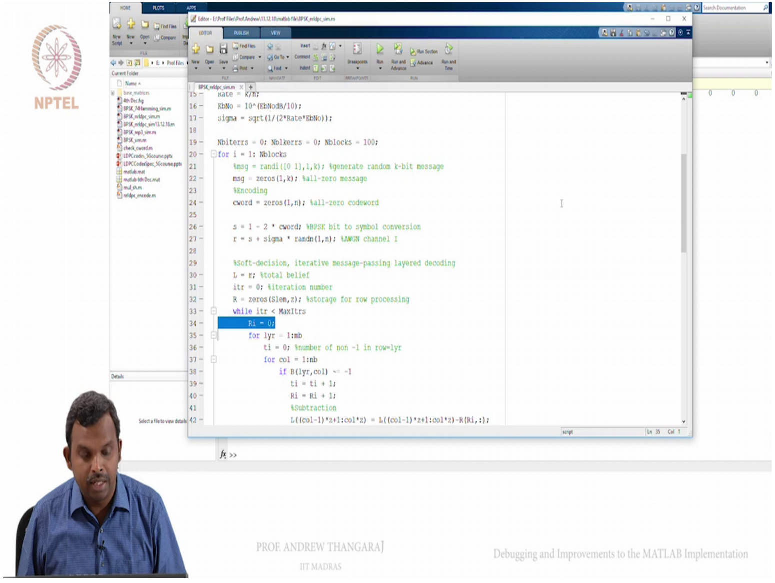
```
    treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

    k = (nb-mb)*z; %number of message bits
    n = nb*z; %number of codeword bits

    Rate = k/n;
    EbNo = 10^(EbNodB/10);
    sigma = sqrt(1/(2*Rate*EbNo));

    Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
    for i = 1: Nblocks
        %msg = randi([0 1],1,k); %generate random k-bit message
        msg = zeros(1,k); %all-zero message
        %Encoding
        cword = zeros(1,n); %all-zero codeword

        s = 1 - 2 * cword; %BPSK bit to symbol conversion
        r = s + sigma * randn(1,n); %AWGN channel I

        %Soft-decision, iterative message-passing layered decoding
        L = r; %total belief
        itr = 0; %iteration number
        R = zeros(Slen,z); %storage for row processing
        while itr < MaxItrs
            Ri = 0;
            for lyr = 1:mb
                ti = 0; %number of non -1 in row=lyr
                for col = 1:nb
```

Editor - E:\Prof Files\Prof.Andrew\13.12.18\matlab file\BPSK_nrldpc_sim.m
EDITOR
PUBLISH
VIEW
BPSK_nrldpc_sim.m
NPTEL
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)



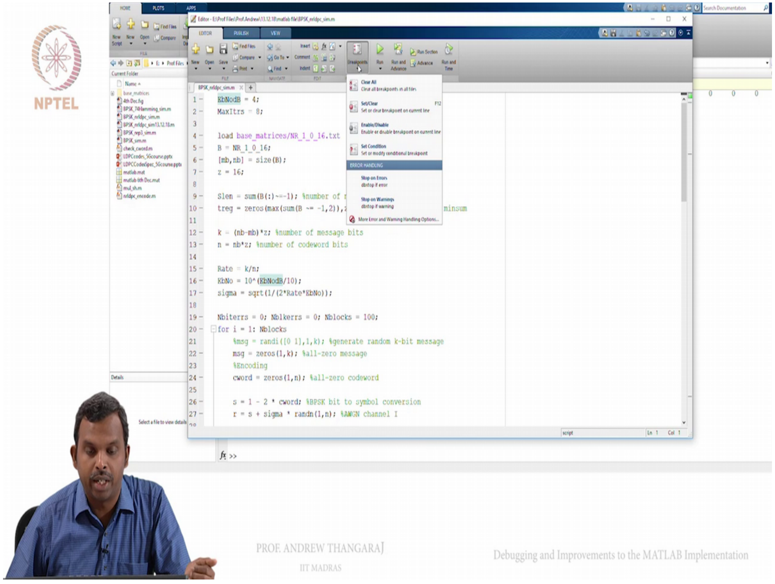
```
    Rate = k/n;
    EbNo = 10^(EbNodB/10);
    sigma = sqrt(1/(2*Rate*EbNo));

    Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
    for i = 1: Nblocks
        %msg = randi([0 1],1,k); %generate random k-bit message
        msg = zeros(1,k); %all-zero message
        %Encoding
        cword = zeros(1,n); %all-zero codeword

        s = 1 - 2 * cword; %BPSK bit to symbol conversion
        r = s + sigma * randn(1,n); %AWGN channel I

        %Soft-decision, iterative message-passing layered decoding
        L = r; %total belief
        itr = 0; %iteration number
        R = zeros(Slen,z); %storage for row processing
        while itr < MaxItrs
            Ri = 0;
            for lyr = 1:mb
                ti = 0; %number of non -1 in row=lyr
                for col = 1:nb
                    if B(lyr,col) ~= -1
                        ti = ti + 1;
                        Ri = Ri + 1;
                        %Subtraction
                        L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
```

Editor - E:\Prof Files\Prof.Andrew\13.12.18\matlab file\BPSK_nrldpc_sim.m
EDITOR
PUBLISH
VIEW
BPSK_nrldpc_sim.m
NPTEL
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

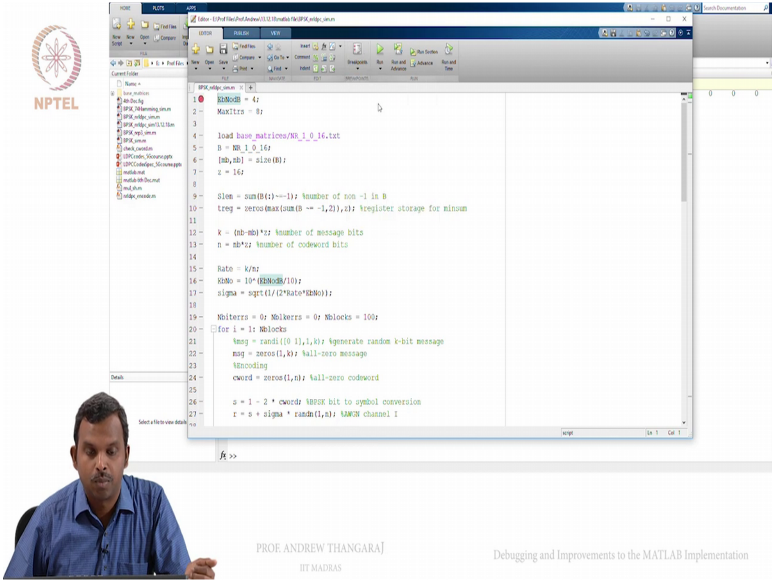
(Refer Slide Time: 01:39)


Editor - E:\Prof Files\Prof Andrew\13.12.18\matlab file\BPSK_nrldpc_sim.m
EDITOR
PUBLISH
VIEW
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
BPSK_nrldpc_sim.m
Clear All
Clear all breakpoints in all files
Set/Clear
Set or clear breakpoint on current line
F12
Enable/Disable
Enable or disable breakpoint on current line
Set Condition
Set or modify conditional breakpoint
ERROR HANDLING
Stop on Errors
dbstop if error
Stop on Warnings
dbstop if warning
More Error and Warning Handling Options...
NPTEL
PROF. ANDREW THANGARAJ
IIT MADRAS
Debugging and Improvements to the MATLAB Implementation

```
EbNodB = 4;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of r
treg = zeros(max(sum(B ~= -1,2)),z                                  minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

breakpoint here,

(Refer Slide Time: 01:41)



```
EbNodB = 4;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

Ok.

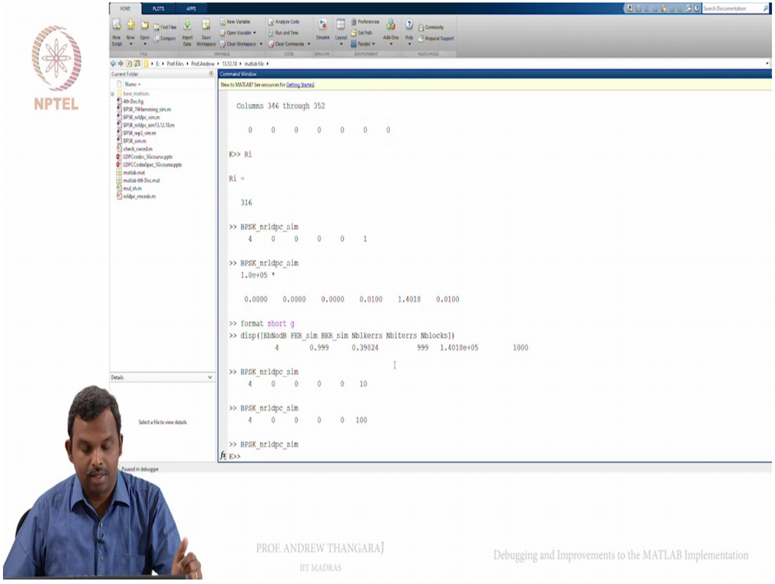
So this is something important. You will single step-through the code to see how it works. And then if you run it, your debugger stopped it there and then you can start single stepping, Ok.

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)



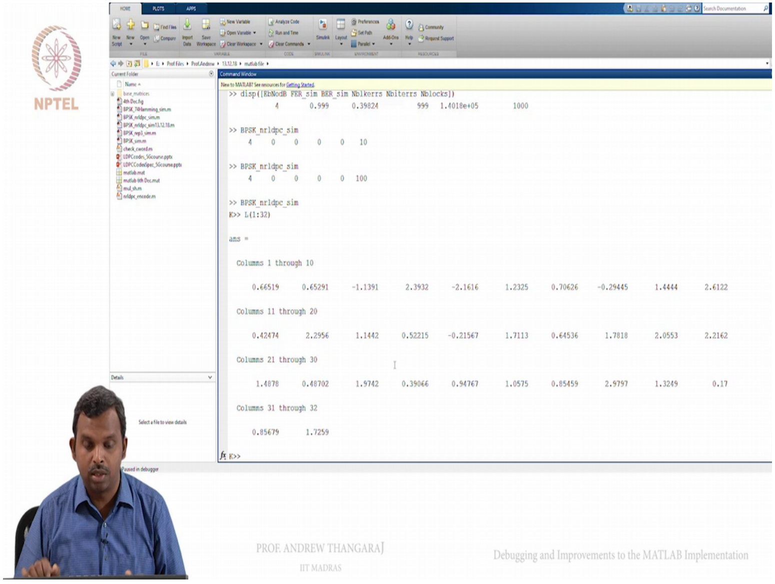
```
  Columns 346 through 352

     0     0     0     0     0     0     0

K>> Ri

Ri =

   316

>> BPSK_nrldpc_sim
     4     0     0     0     0     1

>> BPSK_nrldpc_sim
   1.0e+05 *

    0.0000    0.0000    0.0000    0.0100    1.4018    0.0100

>> format short g
>> disp([EbNodB FER_sim BER_sim Nblkerrs Nbiterrs Nblocks])
            4        0.999      0.39824          999   1.4018e+05         1000

>> BPSK_nrldpc_sim
     4     0     0     0     0    10

>> BPSK_nrldpc_sim
     4     0     0     0     0   100

>> BPSK_nrldpc_sim
K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

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)



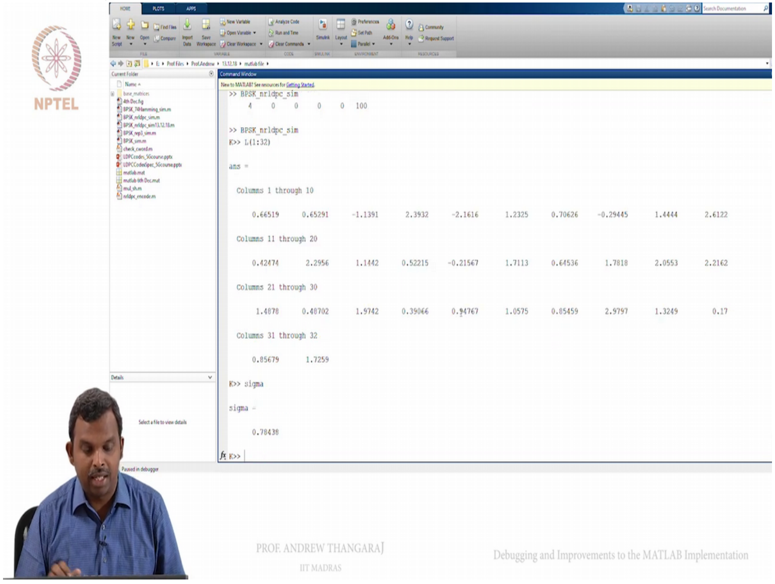
```
>> disp([EbNodB FER_sim BER_sim Nblkerrs Nbiterrs Nblocks])
            4        0.999      0.39824          999   1.4018e+05         1000

>> BPSK_nrldpc_sim
     4     0     0     0     0    10

>> BPSK_nrldpc_sim
     4     0     0     0     0   100

>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

     0.66519      0.65291      -1.1391       2.3932      -2.1616       1.2325      0.70626     -0.29445       1.4444       2.6122

  Columns 11 through 20

     0.42474       2.2956       1.1442      0.52215     -0.21567       1.7113      0.64536       1.7818       2.0553       2.2162

  Columns 21 through 30

      1.4878      0.48702       1.9742      0.39066      0.94767       1.0575      0.85459       2.9797       1.3249         0.17

  Columns 31 through 32

     0.85679       1.7259

K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation blocks so those are the received vectors. Remember we transmitted the all zero codeword, right. So anything positive is no error. Anything negative is error, Ok.

So we sent error plus 1 or B P S K is plus 1 or minus 1and we added Gaussian noise. And you can look at the sigma, sigma will be fairly large,

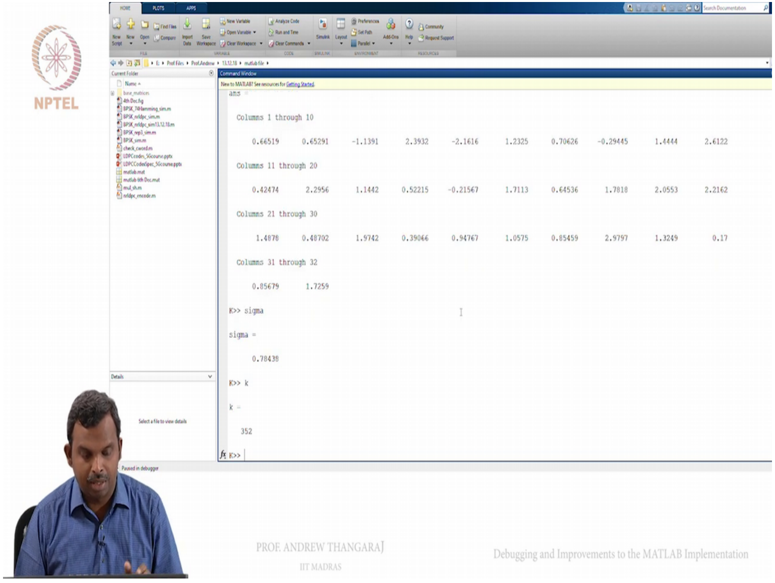
(Refer Slide Time: 02:44)



```
>> BPSK_nrldpc_sim
     4     0     0     0     0   100

>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>>
```

Ok so some point 7 8 because remember the code rate is really, really low, right, so if you remember k, k will be something

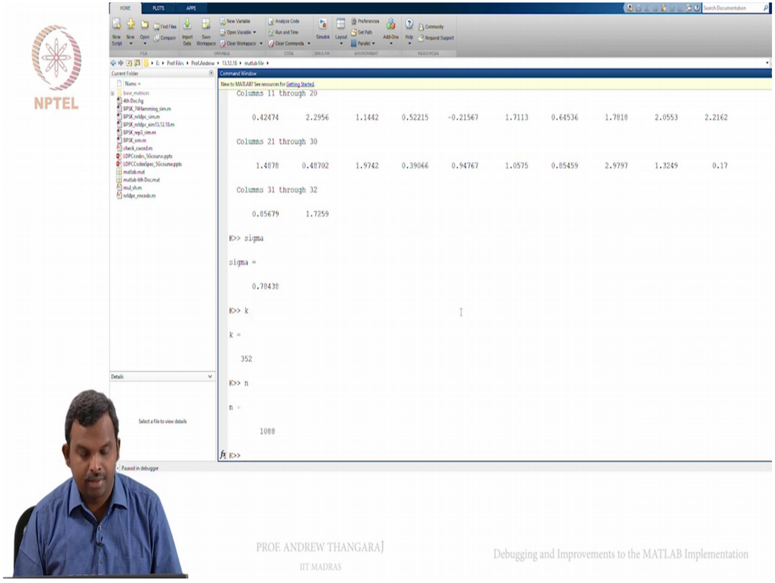
(Refer Slide Time: 02:52)



```
ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =

   352

K>>
```

like 352,

(Refer Slide Time: 02:54)



```
  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =

   352

K>> n

n =

        1088

fx K>>
```

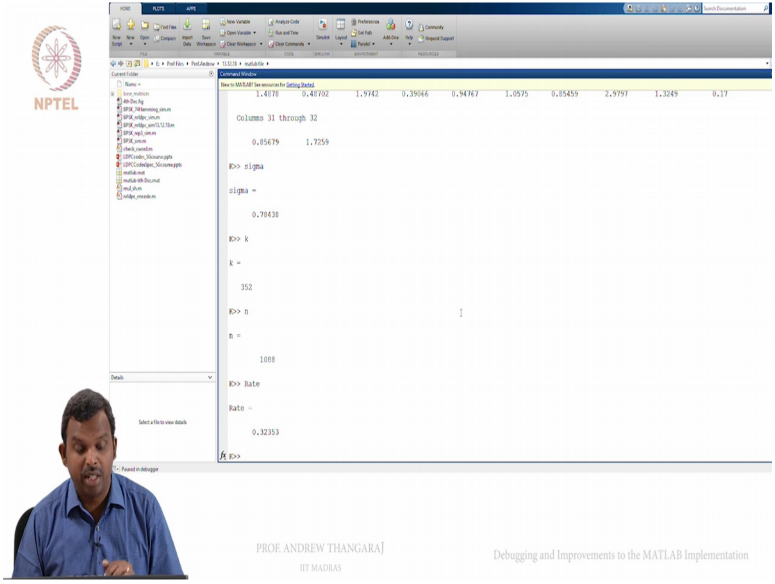
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =

   352

K>> n

n =

        1088

K>> Rate

Rate =

    0.32353

fx K>>
```

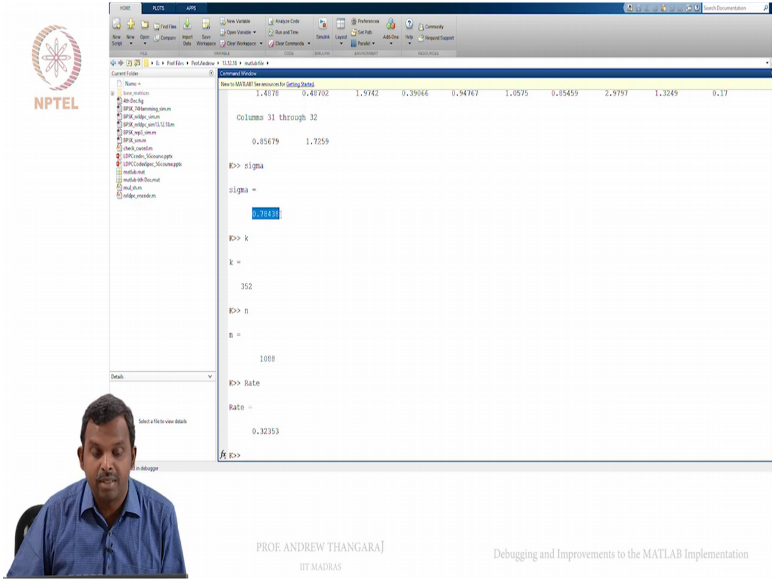
PROF. ANDREW THANGARAJ
IIT MADRAS

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 over N naught of 4 d B and all is quite high E b over N naught, but still the noise will be quite low, quite high. Sigma is point 7 8 4 8.

(Refer Slide Time: 03:29)



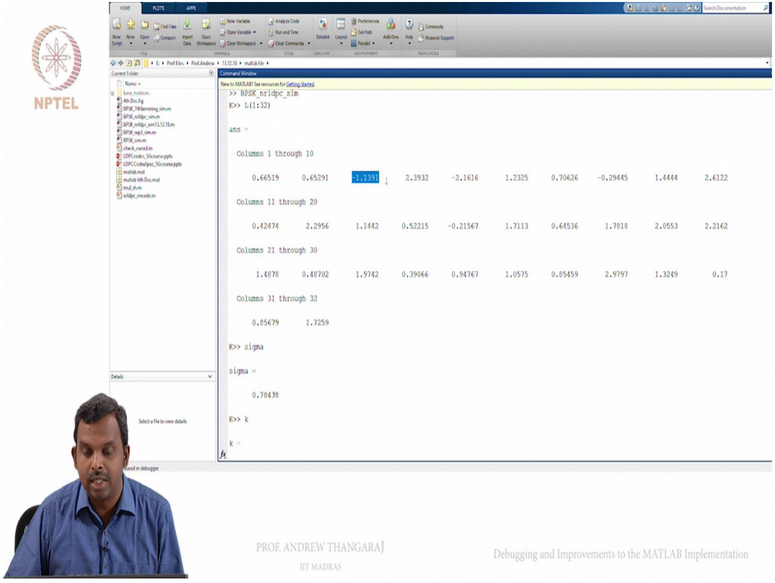
```
       1.4878     0.48702      1.9742     0.39066     0.94767      1.0575     0.85459      2.9797      1.3249        0.17

  Columns 31 through 32

      0.85679      1.7259

K>> sigma

sigma =

      0.78438

K>> k

k =

   352

K>> n

n =

        1088

K>> Rate

Rate =

      0.32353

K>>
```

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)



```
>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

      0.66519     0.65291     -1.1391      2.3932     -2.1616      1.2325     0.70626    -0.29445      1.4444      2.6122

  Columns 11 through 20

      0.42474      2.2956      1.1442     0.52215    -0.21567      1.7113     0.64536      1.7818      2.0553      2.2162

  Columns 21 through 30

       1.4878     0.48702      1.9742     0.39066     0.94767      1.0575     0.85459      2.9797      1.3249        0.17

  Columns 31 through 32

      0.85679      1.7259

K>> sigma

sigma =

      0.78438

K>> k

k =
```

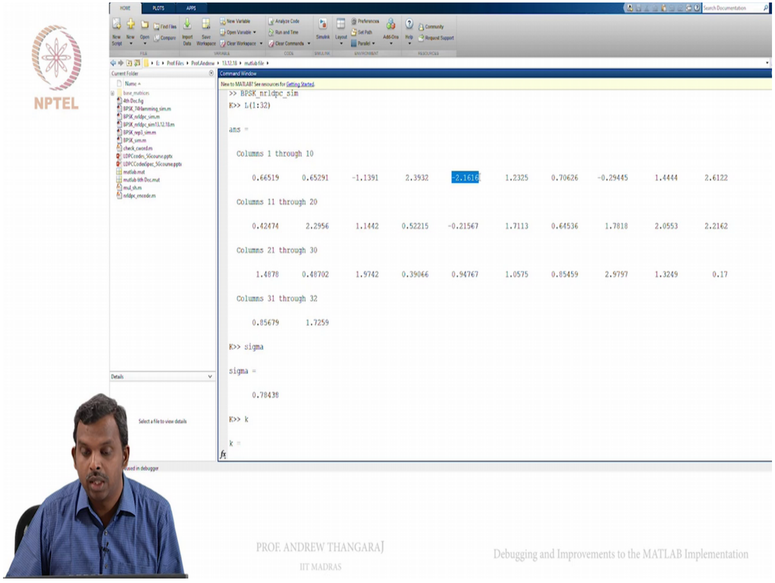
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

received words are in error.

Ok, if you look at this minus 2,

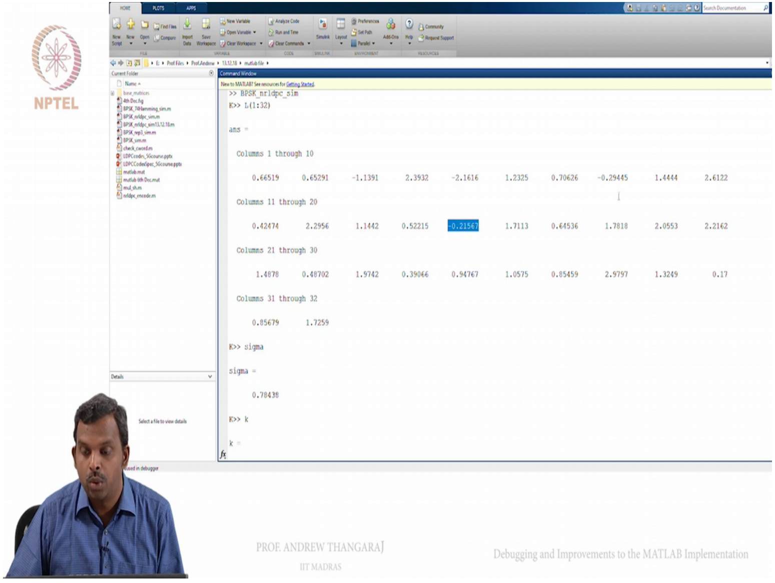
(Refer Slide Time: 03:37)



```
>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =
```

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

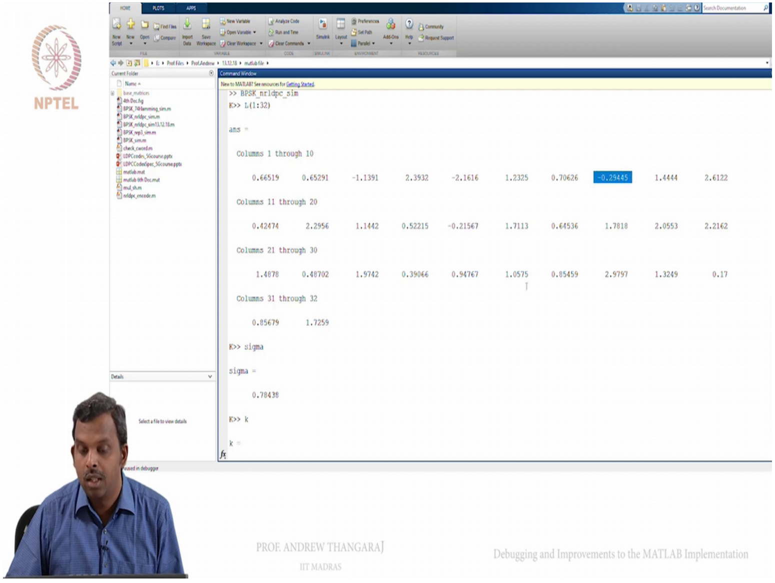
(Refer Slide Time: 03:39)



```
>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =
```

minus point 9,

(Refer Slide Time: 03:41)



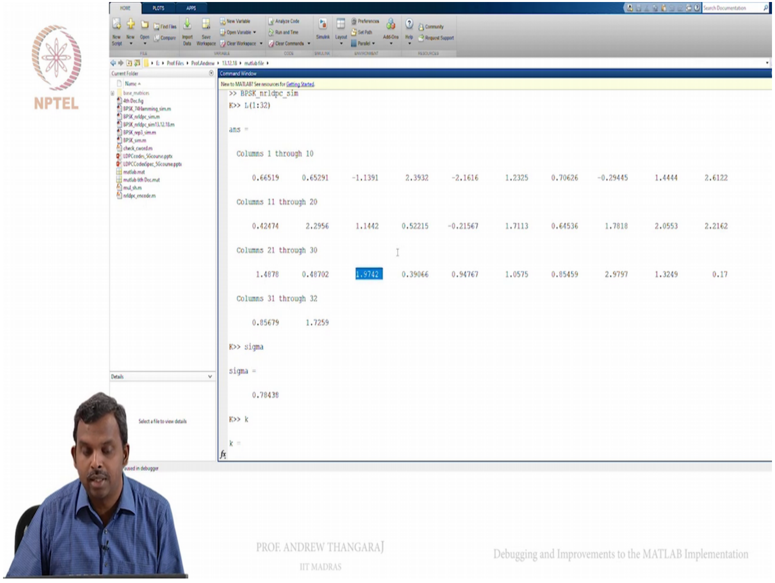
```
>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

quite a few things are in error. But few things are not in error on the other side

(Refer Slide Time: 03:44)



```
>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 20

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.64536     1.7818     2.0553     2.2162

  Columns 21 through 30

     1.4878    0.48702     1.9742    0.39066    0.94767     1.0575    0.85459     2.9797     1.3249       0.17

  Columns 31 through 32

    0.85679     1.7259

K>> sigma

sigma =

    0.78438

K>> k

k =
```

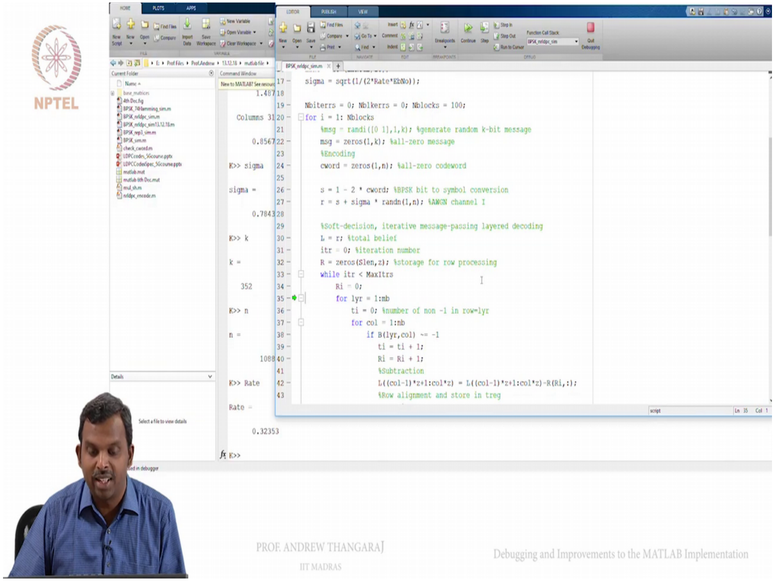
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

also. But you have errors, Ok. So clearly there are errors. And one needs to see how this can get corrected as we run the iterations, Ok.

So if you run more and more iterations it should

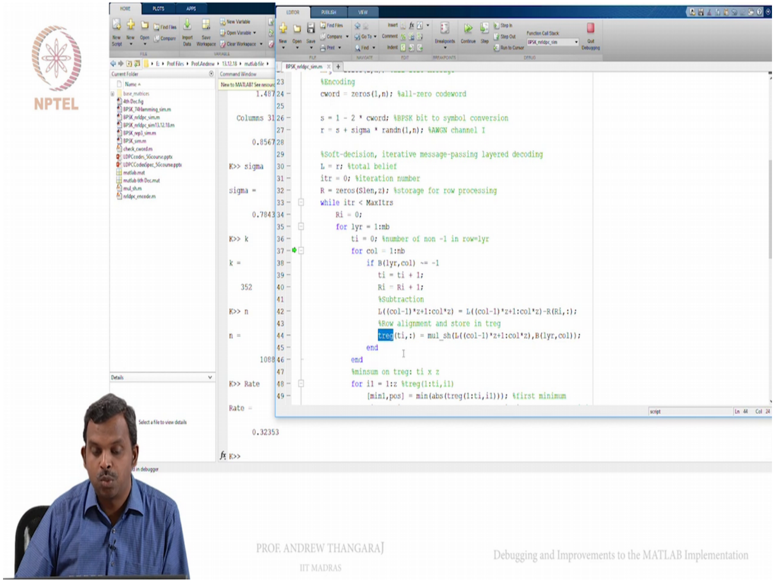
(Refer Slide Time: 03:56)


```
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I

    %Soft-decision, iterative message-passing layered decoding
    L = r; %total belief
    itr = 0; %iteration number
    R = zeros(Slen,z); %storage for row processing
    while itr < MaxItrs
        Ri = 0;
        for lyr = 1:mb
            ti = 0; %number of non -1 in row=lyr
            for col = 1:nb
                if B(lyr,col) ~= -1
                    ti = ti + 1;
                    Ri = Ri + 1;
                    %Subtraction
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                    %Row alignment and store in treg
```
```
      1.487

  Columns 31

      0.8567

>> sigma

sigma =

    0.7843

>> k

k =

   352

>> n

n =

        1088

>> Rate

Rate =

    0.32353

fx >>
```

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

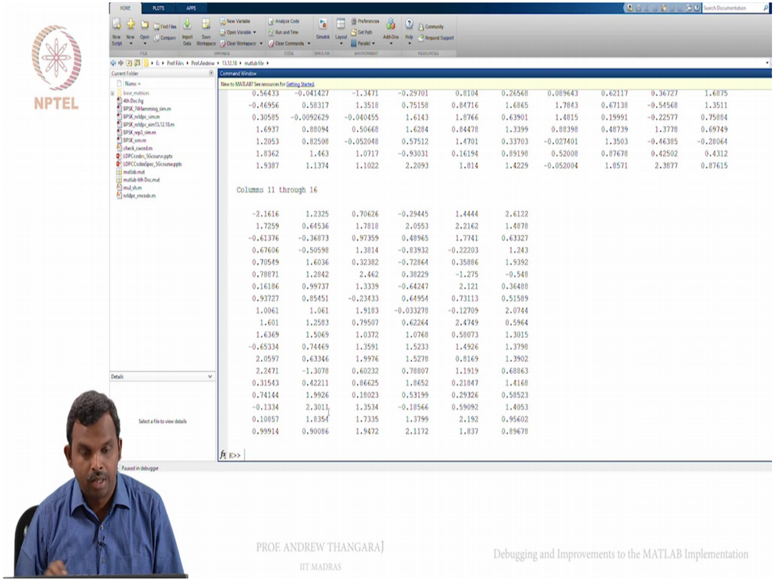
(Refer Slide Time: 04:21)


```
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I

    %Soft-decision, iterative message-passing layered decoding
    L = r; %total belief
    itr = 0; %iteration number
    R = zeros(Slen,z); %storage for row processing
    while itr < MaxItrs
        Ri = 0;
        for lyr = 1:mb
            ti = 0; %number of non -1 in row=lyr
            for col = 1:nb
                if B(lyr,col) ~= -1
                    ti = ti + 1;
                    Ri = Ri + 1;
                    %Subtraction
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                    %Row alignment and store in treg
                    treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
                end
            end
            %minsum on treg: ti x z
            for i1 = 1:z %treg(1:ti,i1)
                [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
```
```
      1.487

  Columns 31

      0.8567

>> sigma

sigma =

    0.7843

>> k

k =

   352

>> n

n =

        1088

>> Rate

Rate =

    0.32353

fx >>
```

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)

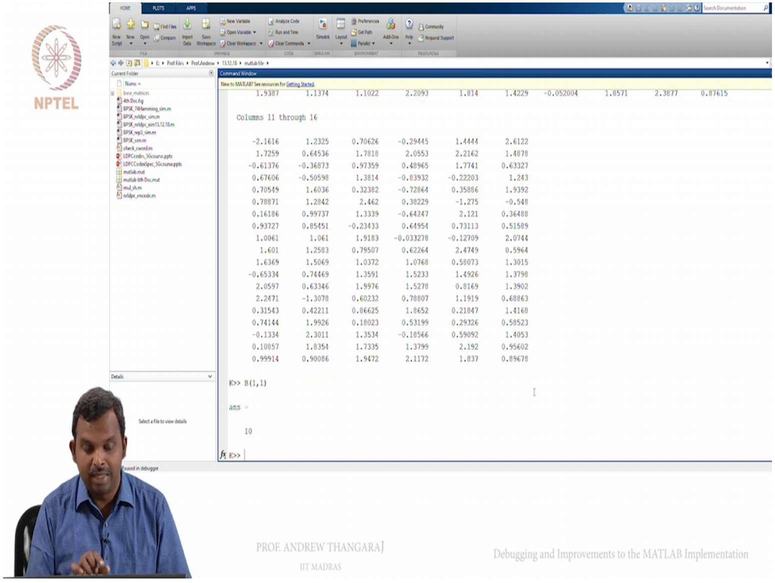

NPTEL
Command Window
New to MATLAB? See resources for Getting Started.
PROF. ANDREW THANGARAJ
IIT MADRAS
Debugging and Improvements to the MATLAB Implementation

```
    0.56433   -0.041427     -1.3471    -0.29701      0.8104     0.26568    0.089643     0.62117     0.36727      1.6875
   -0.46956     0.58317      1.3518     0.75158     0.84716      1.6865      1.7843     0.67138    -0.54568      1.3511
    0.30585  -0.0092629   -0.040455      1.6143      1.8766     0.63901      1.4815     0.19991    -0.22577     0.75884
     1.6937     0.88094     0.50668      1.6284     0.84478      1.3399     0.88398     0.48739      1.3778     0.69749
     1.2053     0.82508   -0.052048     0.57512      1.4701     0.33703   -0.027401      1.3503    -0.46385    -0.28064
     1.8362       1.463      1.0717    -0.93031     0.16194     0.89198     0.52008     0.87678     0.42502      0.4312
     1.9387      1.1374      1.1022      2.2093       1.814      1.4229   -0.052004      1.8571      2.3877     0.87615

  Columns 11 through 16

    -2.1616      1.2325     0.70626    -0.29445      1.4444      2.6122
     1.7259     0.64536      1.7818      2.0553      2.2162      1.4878
   -0.61376    -0.36873     0.97359     0.48965      1.7741     0.63327
    0.67606    -0.50598      1.3814    -0.83932    -0.22203       1.243
    0.70549      1.6036     0.32382    -0.72864     0.35886      1.9392
    0.78871      1.2842       2.462     0.38229      -1.275      -0.548
    0.16186     0.99737      1.3339    -0.64247       2.121     0.36488
    0.93727     0.85451    -0.23433     0.64954     0.73113     0.51589
     1.0061       1.061      1.9183   -0.033278    -0.12709      2.0744
      1.601      1.2583     0.79507     0.62264      2.4749      0.5964
     1.6369      1.5069      1.0372      1.0768     0.58073      1.3015
   -0.65334     0.74469      1.3591      1.5233      1.4926      1.3798
     2.0597     0.63346      1.9976      1.5278      0.8169      1.3902
     2.2471     -1.3078     0.60232     0.78807      1.1919     0.68863
    0.31543     0.42211     0.86625      1.8652     0.21847      1.4168
    0.74144      1.9926     0.18023     0.53199     0.29326     0.58523
    -0.1334      2.3011      1.3534    -0.18566     0.59092      1.4053
    0.10857      1.8354      1.7335      1.3799       2.192     0.95602
    0.99914     0.90086      1.9472      2.1172       1.837     0.89678

K>>
```

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,

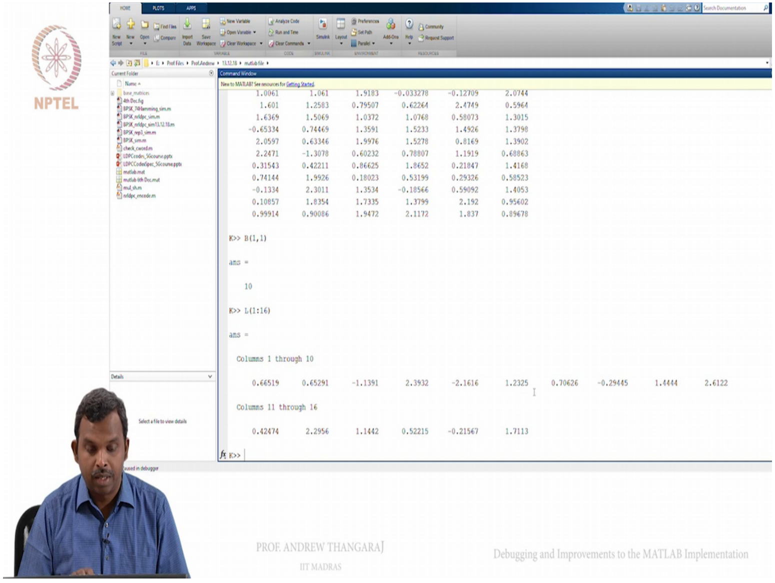
(Refer Slide Time: 04:52)


NPTEL
Command Window
New to MATLAB? See resources for Getting Started.
PROF. ANDREW THANGARAJ
IIT MADRAS
Debugging and Improvements to the MATLAB Implementation

```
     1.9387      1.1374      1.1022      2.2093       1.814      1.4229   -0.052004      1.8571      2.3877     0.87615

  Columns 11 through 16

    -2.1616      1.2325     0.70626    -0.29445      1.4444      2.6122
     1.7259     0.64536      1.7818      2.0553      2.2162      1.4878
   -0.61376    -0.36873     0.97359     0.48965      1.7741     0.63327
    0.67606    -0.50598      1.3814    -0.83932    -0.22203       1.243
    0.70549      1.6036     0.32382    -0.72864     0.35886      1.9392
    0.78871      1.2842       2.462     0.38229      -1.275      -0.548
    0.16186     0.99737      1.3339    -0.64247       2.121     0.36488
    0.93727     0.85451    -0.23433     0.64954     0.73113     0.51589
     1.0061       1.061      1.9183   -0.033278    -0.12709      2.0744
      1.601      1.2583     0.79507     0.62264      2.4749      0.5964
     1.6369      1.5069      1.0372      1.0768     0.58073      1.3015
   -0.65334     0.74469      1.3591      1.5233      1.4926      1.3798
     2.0597     0.63346      1.9976      1.5278      0.8169      1.3902
     2.2471     -1.3078     0.60232     0.78807      1.1919     0.68863
    0.31543     0.42211     0.86625      1.8652     0.21847      1.4168
    0.74144      1.9926     0.18023     0.53199     0.29326     0.58523
    -0.1334      2.3011      1.3534    -0.18566     0.59092      1.4053
    0.10857      1.8354      1.7335      1.3799       2.192     0.95602
    0.99914     0.90086      1.9472      2.1172       1.837     0.89678

K>> B(1,1)

ans =

    10

K>>
```

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

(Refer Slide Time: 04:56)



```
    1.0061      1.061     1.9183  -0.033278   -0.12709     2.0744
     1.601     1.2583    0.79507    0.62264     2.4749     0.5964
    1.6369     1.5069     1.0372     1.0768    0.58073     1.3015
  -0.65334    0.74469     1.3591     1.5233     1.4926     1.3798
    2.0597    0.63346     1.9976     1.5278     0.8169     1.3902
    2.2471    -1.3078    0.60232    0.78807     1.1919    0.68863
   0.31543    0.42211    0.86625     1.8652    0.21847     1.4168
   0.74144     1.9926    0.18023    0.53199    0.29326    0.58523
   -0.1334     2.3011     1.3534   -0.18566    0.59092     1.4053
   0.10857     1.8354     1.7335     1.3799      2.192    0.95602
   0.99914    0.90086     1.9472     2.1172      1.837    0.89678

K>> B(1,1)

ans =

    10

K>> L(1:16)

ans =

  Columns 1 through 10

   0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 16

   0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113

K>>
```

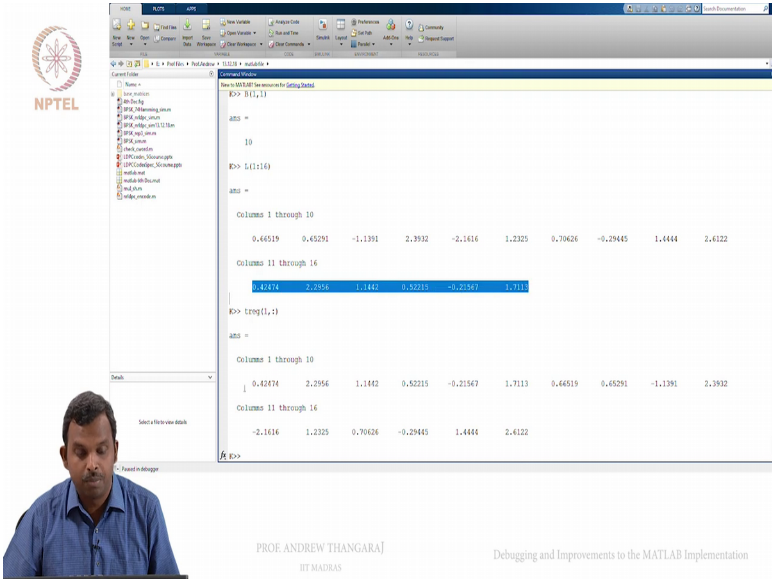
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



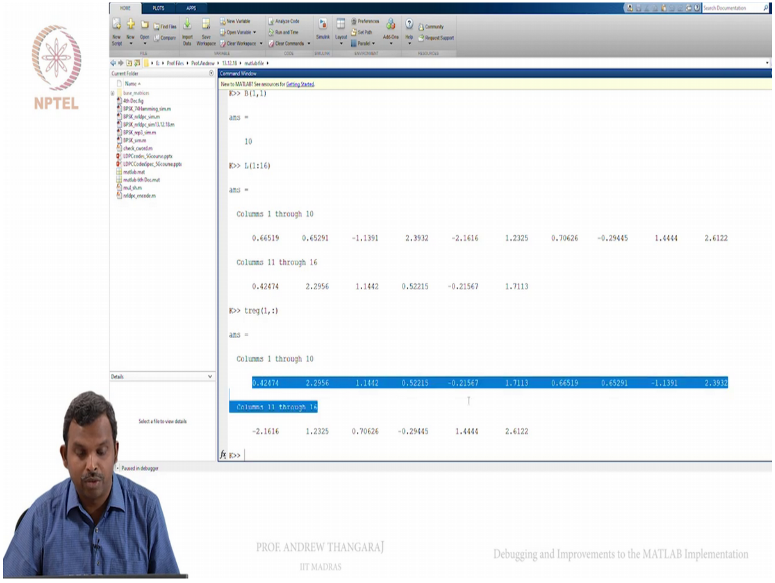
```
K>> B(1,1)

ans =

    10

K>> L(1:16)

ans =

  Columns 1 through 10

   0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 16

   0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113

K>> treg(1,:)

ans =

  Columns 1 through 10

   0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.66519    0.65291    -1.1391     2.3932

  Columns 11 through 16

   -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

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)

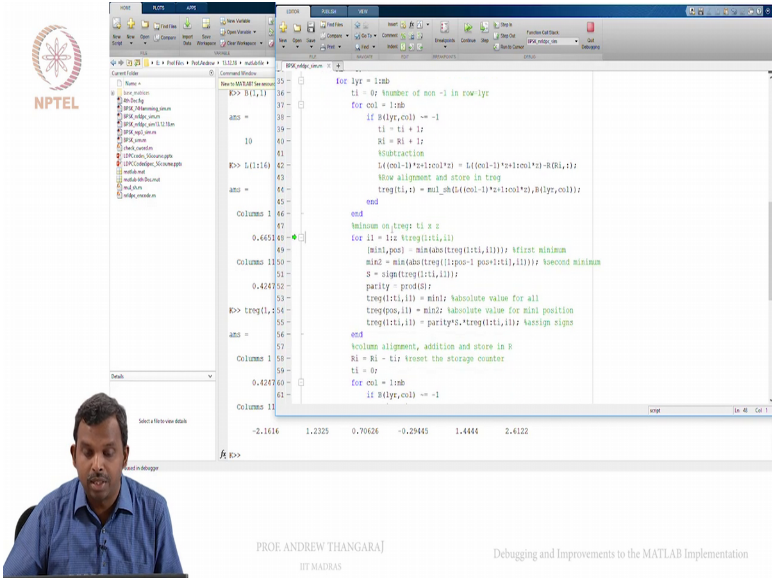


```
K>> B(1,1)

ans =

    10

K>> L(1:16)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 16

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113

K>> treg(1,:)

ans =

  Columns 1 through 10

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.66519    0.65291    -1.1391     2.3932

  Columns 11 through 16

    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

K>>
```

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)



```
for lyr = 1:mb
    ti = 0; %number of non -1 in row=lyr
    for col = 1:nb
        if B(lyr,col) ~= -1
            ti = ti + 1;
            Ri = Ri + 1;
            %Subtraction
            L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
            %Row alignment and store in treg
            treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
        end
    end
    %minsum on treg: ti x z
    for i1 = 1:z %treg(1:ti,i1)
        [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
        min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
        S = sign(treg(1:ti,i1));
        parity = prod(S);
        treg(1:ti,i1) = min1; %absolute value for all
        treg(pos,i1) = min2; %absolute value for min1 position
        treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
    end
    %column alignment, addition and store in R
    Ri = Ri - ti; %reset the storage counter
    ti = 0;
    for col = 1:nb
        if B(lyr,col) ~= -1
```

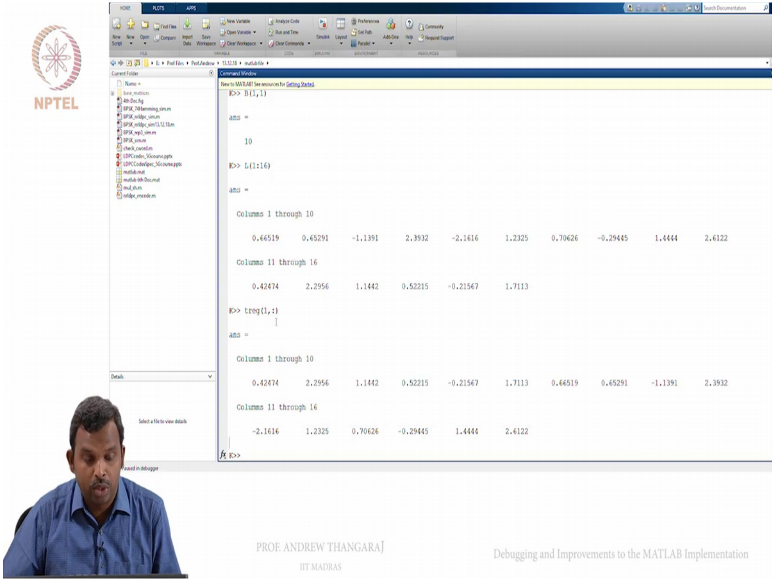
```
    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

K>>
```

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)



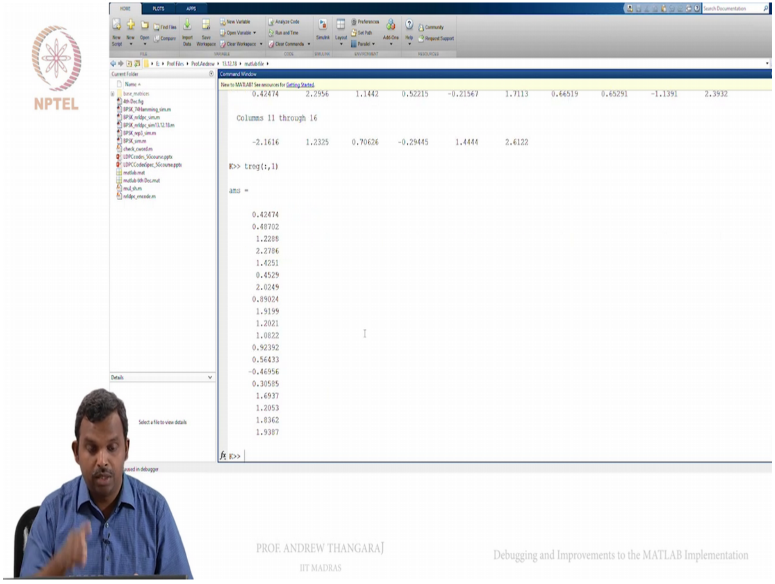
```
K>> B(1,1)

ans =

    10

K>> L(1:16)

ans =

  Columns 1 through 10

    0.66519    0.65291    -1.1391     2.3932    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

  Columns 11 through 16

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113

K>> treg(1,:)

ans =

  Columns 1 through 10

    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.66519    0.65291    -1.1391     2.3932

  Columns 11 through 16

    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

K>>
```

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)



```
    0.42474     2.2956     1.1442    0.52215   -0.21567     1.7113    0.66519    0.65291    -1.1391     2.3932

  Columns 11 through 16

    -2.1616     1.2325    0.70626   -0.29445     1.4444     2.6122

K>> treg(:,1)

ans =

    0.42474
    0.48702
     1.2288
     2.2786
     1.4251
     0.4529
     2.0249
    0.89024
     1.9199
     1.2021
     1.0822
    0.92392
    0.56433
   -0.46956
    0.30585
     1.6937
     1.2053
     1.8362
     1.9387

K>>
```

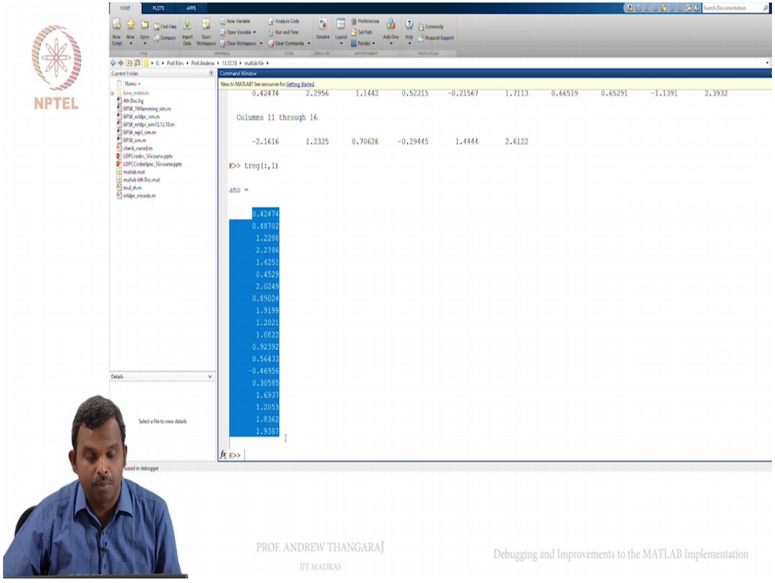
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



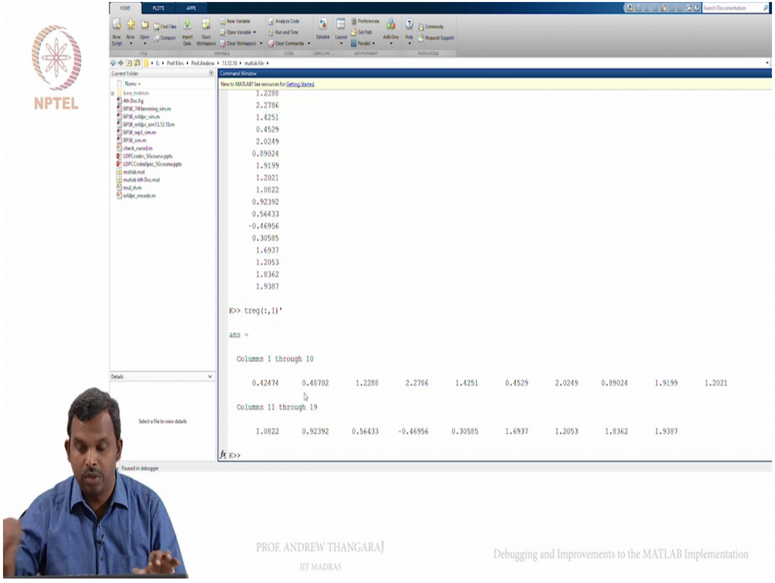
```
    0.42474      2.2956      1.1442     0.52215    -0.21567      1.7113     0.66519     0.65291     -1.1391      2.3932

  Columns 11 through 16

    -2.1616      1.2325     0.70626    -0.29445      1.4444      2.6122

K>> treg(:,1)

ans =

    0.42474
    0.48702
     1.2288
     2.2786
     1.4251
     0.4529
     2.0249
    0.89024
     1.9199
     1.2021
     1.0822
    0.92392
    0.56433
   -0.46956
    0.30585
     1.6937
     1.2053
     1.8362
     1.9387

K>>
```

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)



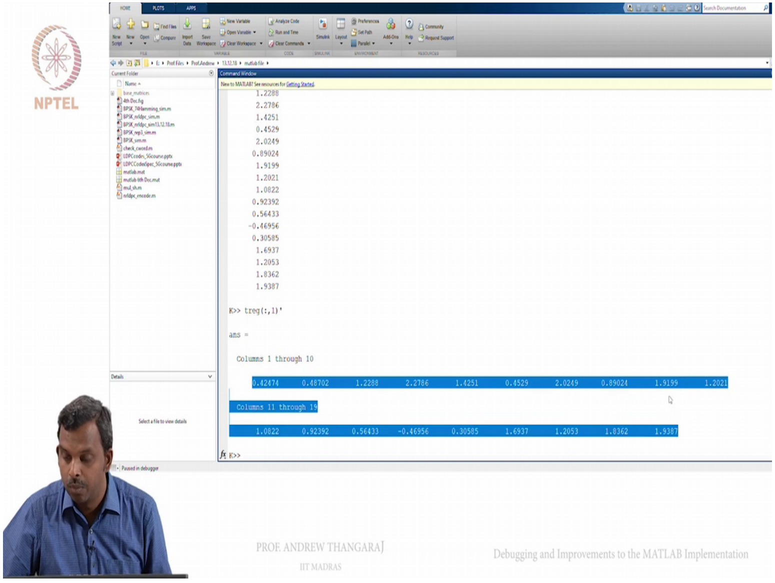
```
     1.2288
     2.2786
     1.4251
     0.4529
     2.0249
    0.89024
     1.9199
     1.2021
     1.0822
    0.92392
    0.56433
   -0.46956
    0.30585
     1.6937
     1.2053
     1.8362
     1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474     0.48702      1.2288      2.2786      1.4251      0.4529      2.0249     0.89024      1.9199      1.2021

  Columns 11 through 19

     1.0822     0.92392     0.56433    -0.46956     0.30585      1.6937      1.2053      1.8362      1.9387

K>>
```

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.

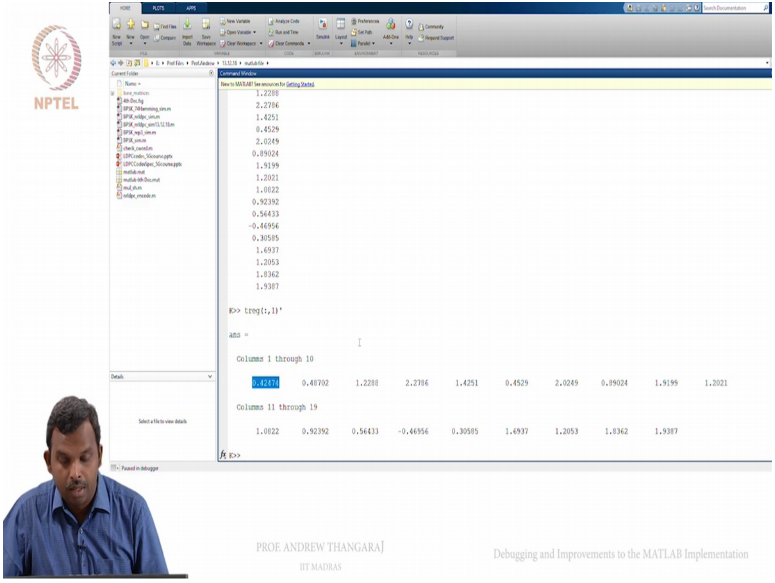
(Refer Slide Time: 06:14)



```
    1.2288
    2.2786
    1.4251
    0.4529
    2.0249
   0.89024
    1.9199
    1.2021
    1.0822
   0.92392
   0.56433
  -0.46956
   0.30585
    1.6937
    1.2053
    1.8362
    1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

   0.42474    0.48702    1.2288    2.2786    1.4251    0.4529    2.0249    0.89024    1.9199    1.2021

  Columns 11 through 19

    1.0822    0.92392    0.56433    -0.46956    0.30585    1.6937    1.2053    1.8362    1.9387

K>>
```

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)

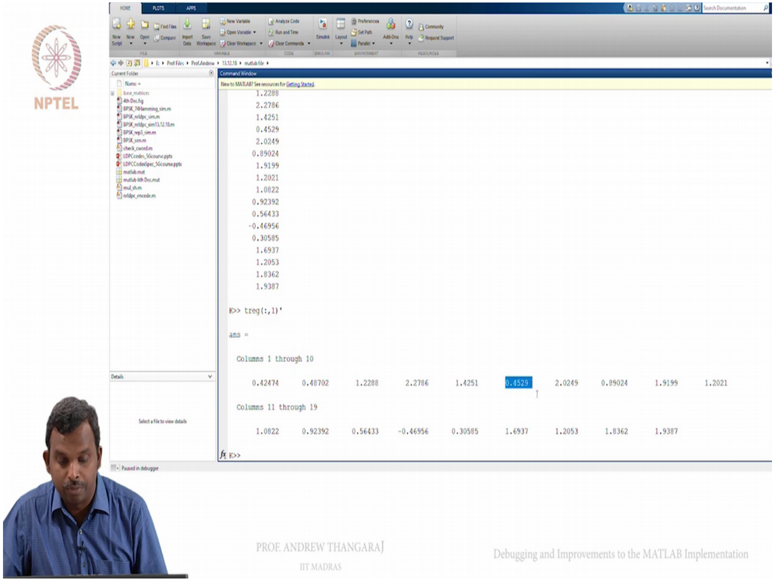


```
    1.2288
    2.2786
    1.4251
    0.4529
    2.0249
   0.89024
    1.9199
    1.2021
    1.0822
   0.92392
   0.56433
  -0.46956
   0.30585
    1.6937
    1.2053
    1.8362
    1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

   0.42474    0.48702    1.2288    2.2786    1.4251    0.4529    2.0249    0.89024    1.9199    1.2021

  Columns 11 through 19

    1.0822    0.92392    0.56433    -0.46956    0.30585    1.6937    1.2053    1.8362    1.9387

K>>
```

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)



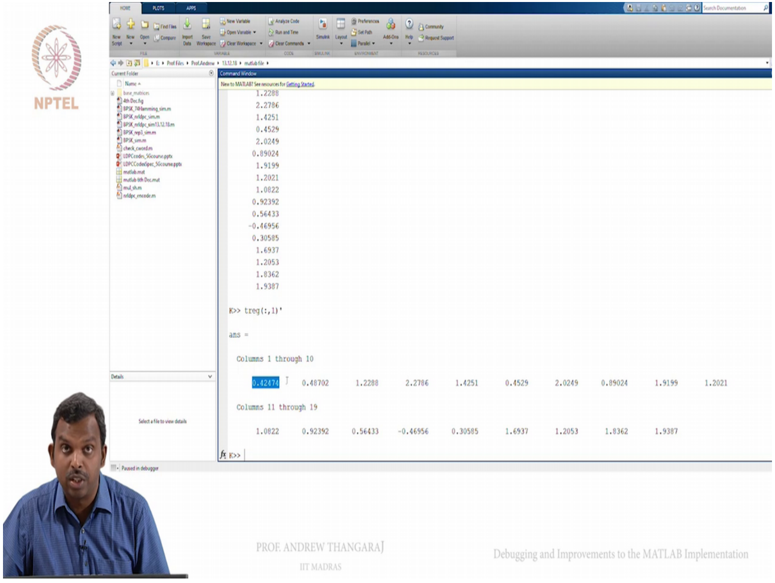
```
   1.2288
   2.2786
   1.4251
   0.4529
   2.0249
  0.89024
   1.9199
   1.2021
   1.0822
  0.92392
  0.56433
 -0.46956
  0.30585
   1.6937
   1.2053
   1.8362
   1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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

(Refer Slide Time: 06:38)



```
   1.2288
   2.2786
   1.4251
   0.4529
   2.0249
  0.89024
   1.9199
   1.2021
   1.0822
  0.92392
  0.56433
 -0.46956
  0.30585
   1.6937
   1.2053
   1.8362
   1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>>
```

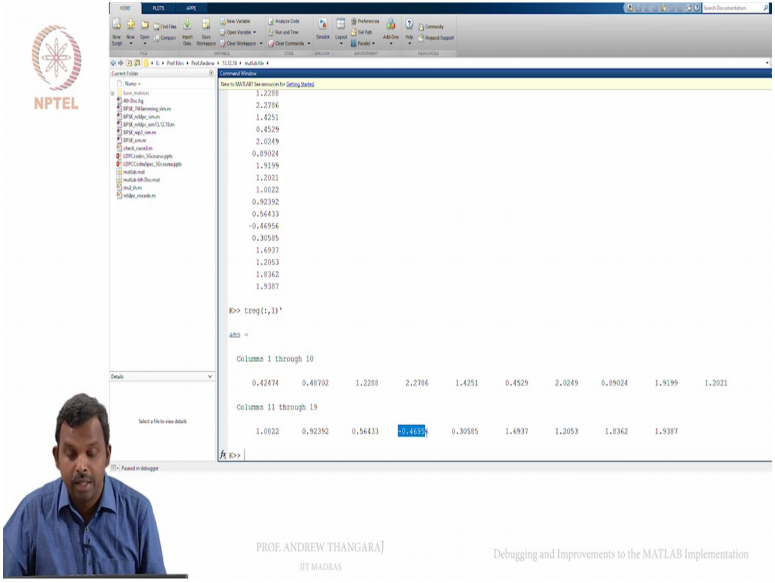
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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)



```
    1.2288
    2.2786
    1.4251
    0.4529
    2.0249
   0.89024
    1.9199
    1.2021
    1.0822
   0.92392
   0.56433
  -0.46956
   0.30585
    1.6937
    1.2053
    1.8362
    1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

   0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

    1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>>
```

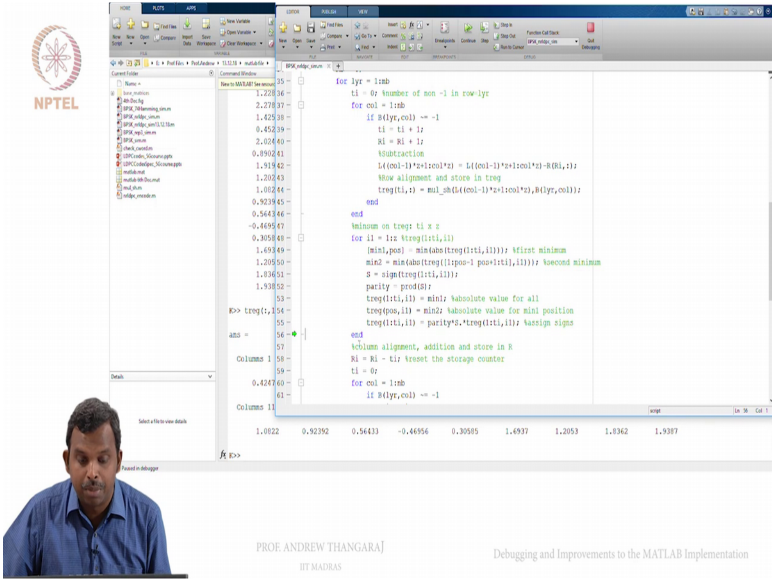
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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)



```
    for lyr = 1:mb
        ti = 0; %number of non -1 in row=lyr
        for col = 1:nb
            if B(lyr,col) ~= -1
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = 1:nb
            if B(lyr,col) ~= -1
```

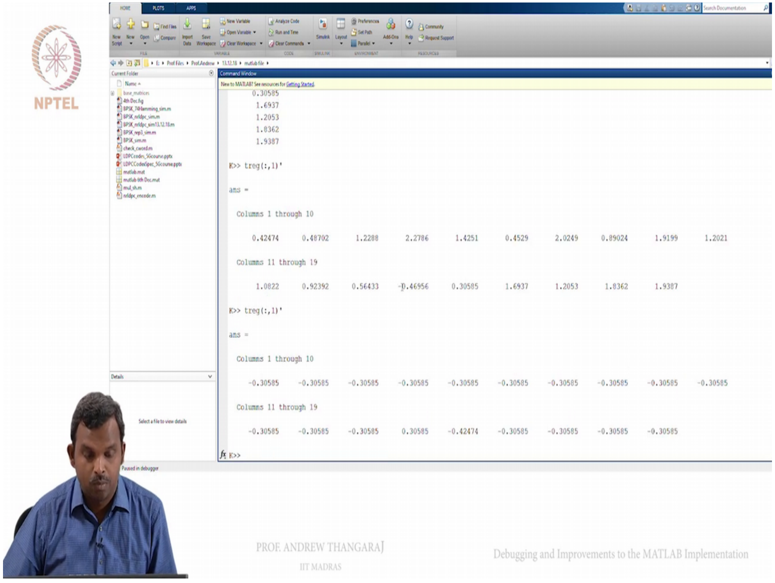
```
    1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

and we see the same thing, right.

(Refer Slide Time: 07:18)



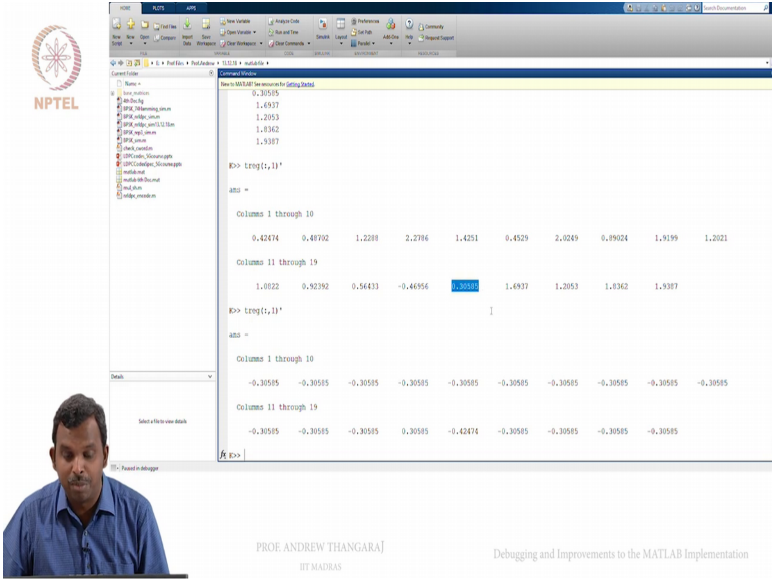
```
New to MATLAB? See resources for Getting Started.
      0.30585
       1.6937
       1.2053
       1.8362
       1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>> treg(:,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

fx K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok alright, so I missed one value, as usual, not very surprising. It looks like point 3 0 5 8 5

(Refer Slide Time: 07:29)



```
New to MATLAB? See resources for Getting Started.
      0.30585
       1.6937
       1.2053
       1.8362
       1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>> treg(:,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

fx K>>
```

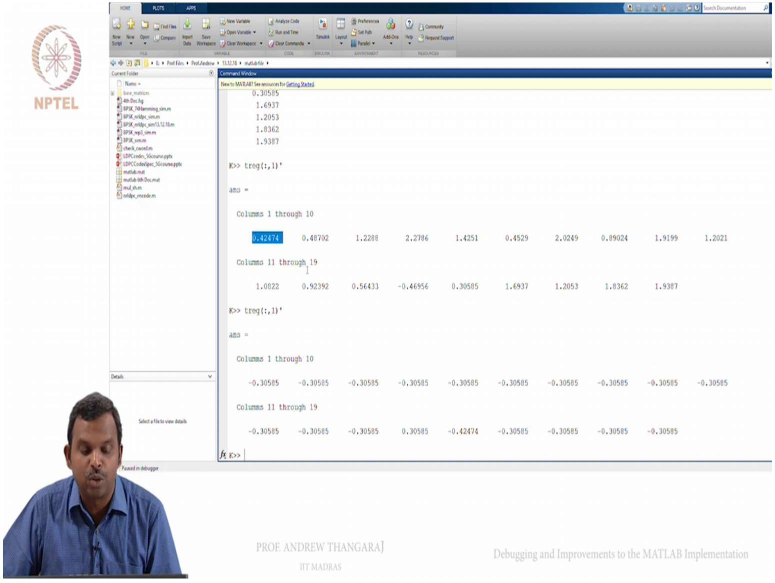
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

is min 1, sorry, Ok. So this can happen if you do it by manually, these kind of errors can happen.

Of course program is not going to make that error. So min 1 I was totally wrong. It is not point 4 2 4 7 4. It was point 3 0 5 8 5, Ok and min 2

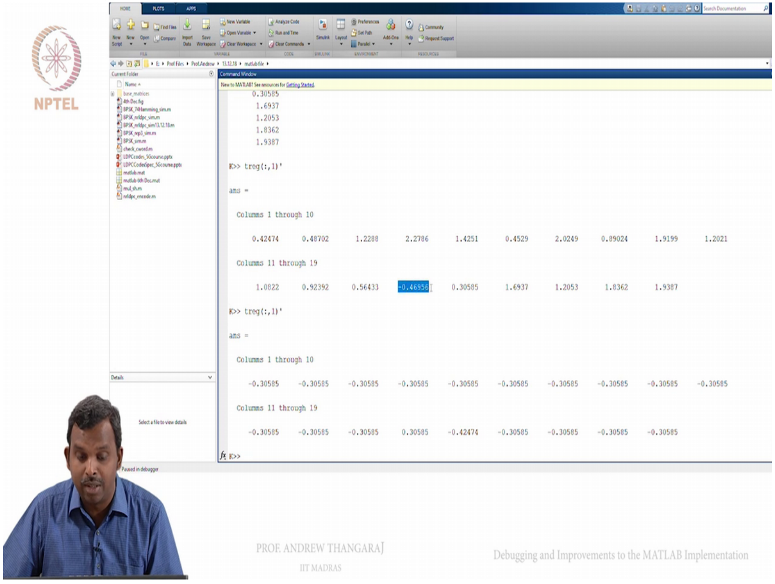
(Refer Slide Time: 07:45)



```
   0.30585
    1.6937
    1.2053
    1.8362
    1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>> treg(:,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

K>>
```

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

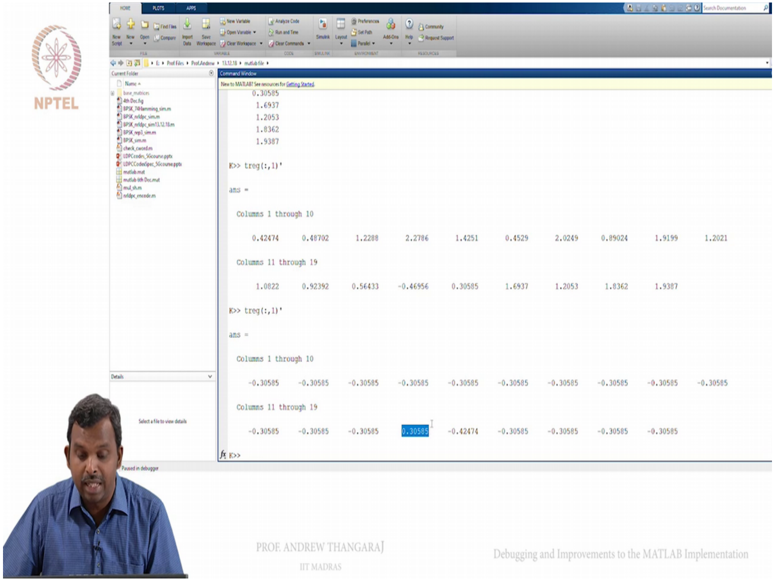
(Refer Slide Time: 07:57)



```
   0.30585
    1.6937
    1.2053
    1.8362
    1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>> treg(:,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

K>>
```

which is positive sign, everything else is

(Refer Slide Time: 07:59)



```
     0.30585
      1.6937
      1.2053
      1.8362
      1.9387

K>> treg(:,1)'

ans =

  Columns 1 through 10

    0.42474    0.48702     1.2288     2.2786     1.4251     0.4529     2.0249    0.89024     1.9199     1.2021

  Columns 11 through 19

     1.0822    0.92392    0.56433   -0.46956    0.30585     1.6937     1.2053     1.8362     1.9387

K>> treg(:,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

fx K>>
```

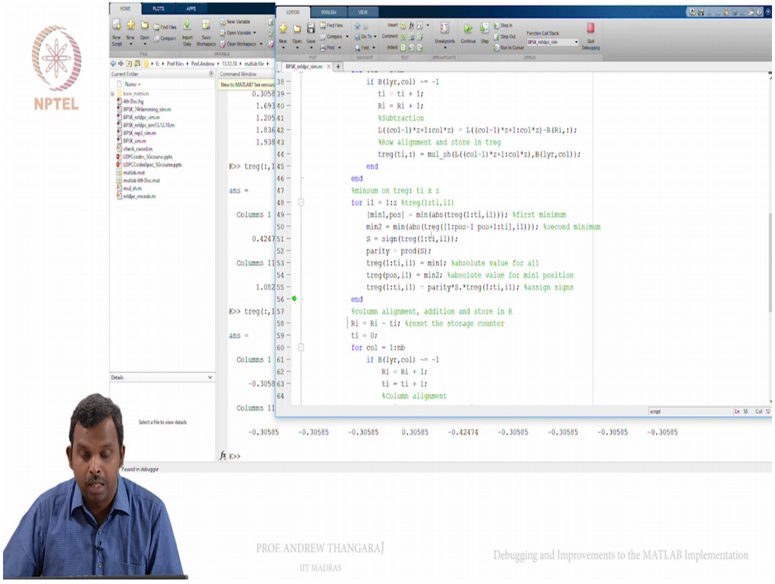
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
        if B(lyr,col) ~= -1
            ti = ti + 1;
            Ri = Ri + 1;
            %Subtraction
            L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
            %Row alignment and store in treg
            treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
        end
    end
    %minsum on treg: ti x z
    for i1 = 1:z %treg(1:ti,i1)
        [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
        min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
        S = sign(treg(1:ti,i1));
        parity = prod(S);
        treg(1:ti,i1) = min1; %absolute value for all
        treg(pos,i1) = min2; %absolute value for min1 position
        treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
    end
    %column alignment, addition and store in R
    Ri = Ri - ti; %reset the storage counter
    ti = 0;
    for col = 1:nb
        if B(lyr,col) ~= -1
            Ri = Ri + 1;
            ti = ti + 1;
            %Column alignment
```

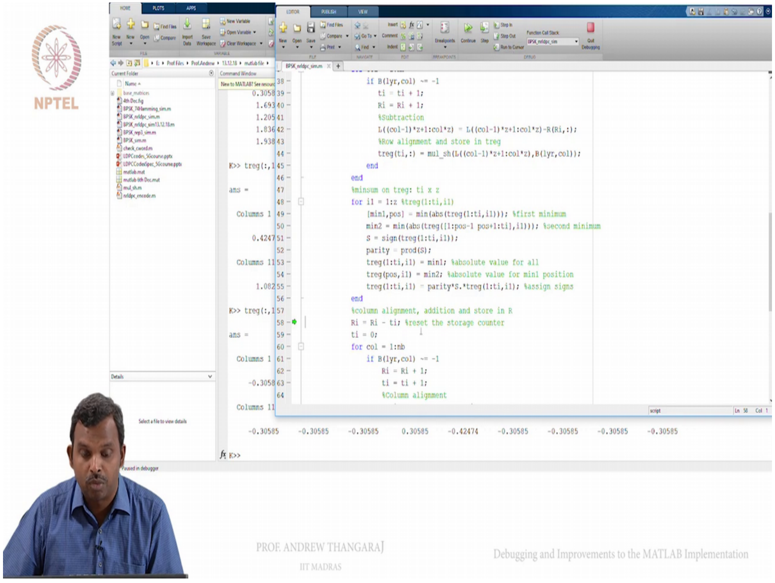
```
   -0.30585   -0.30585   -0.30585    0.30585   -0.42474   -0.30585   -0.30585   -0.30585   -0.30585
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

come out here,

(Refer Slide Time: 08:08)



```
            if B(lyr,col) ~= -1
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = 1:nb
            if B(lyr,col) ~= -1
                Ri = Ri + 1;
                ti = ti + 1;
                %Column alignment
```

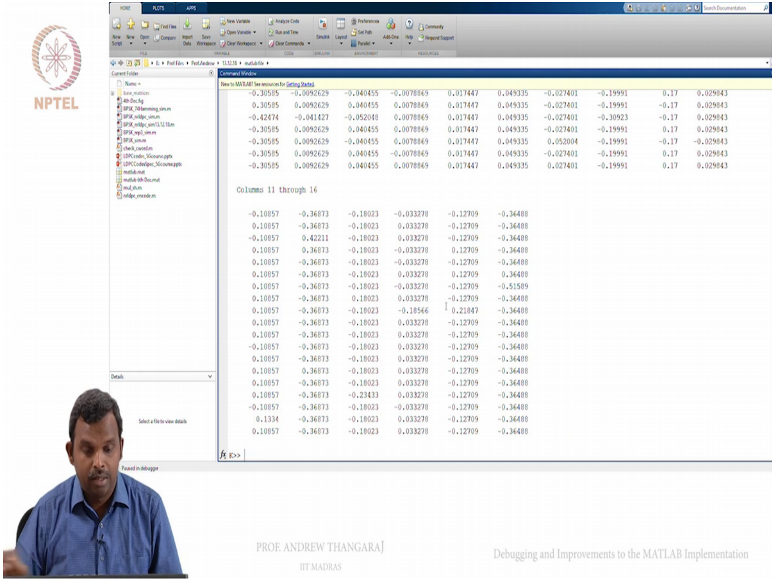
```
  -0.30585   -0.30585   -0.30585    0.30585   -0.42474   -0.30585   -0.30585   -0.30585   -0.30585

fx K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok. And then if you look at t reg you will see

(Refer Slide Time: 08:12)



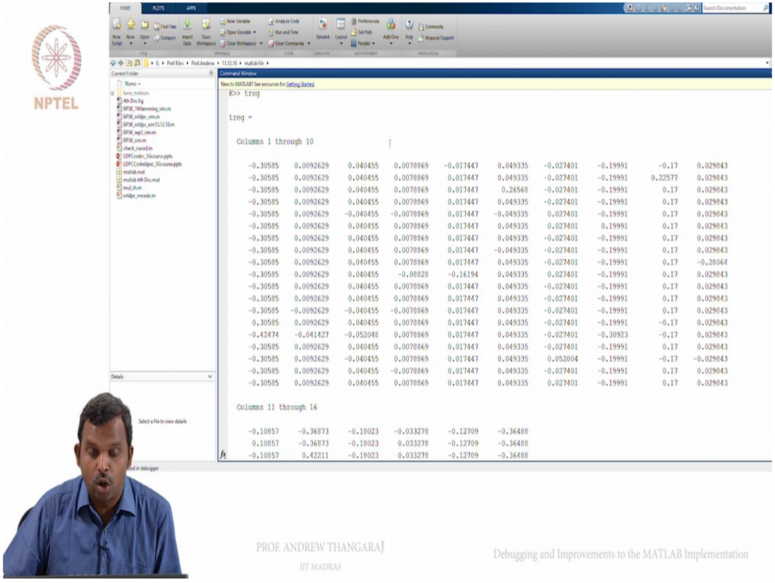
```
  -0.30585  -0.0092629  -0.040455  -0.0078869   0.017447   0.049335  -0.027401   -0.19991      0.17   0.029843
   0.30585   0.0092629   0.040455   0.0078869   0.017447   0.049335  -0.027401   -0.19991     -0.17   0.029843
  -0.42474   -0.041427  -0.052048   0.0078869   0.017447   0.049335  -0.027401   -0.30923     -0.17   0.029843
  -0.30585   0.0092629   0.040455   0.0078869   0.017447   0.049335  -0.027401   -0.19991      0.17   0.029843
  -0.30585   0.0092629  -0.040455   0.0078869   0.017447   0.049335   0.052004   -0.19991     -0.17  -0.029843
  -0.30585   0.0092629   0.040455  -0.0078869   0.017447   0.049335  -0.027401   -0.19991      0.17   0.029843
  -0.30585   0.0092629   0.040455   0.0078869   0.017447   0.049335   0.027401   -0.19991      0.17   0.029843

 Columns 11 through 16

  -0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
  -0.10857    0.42211   -0.18023   0.033278   -0.12709   -0.36488
   0.10857    0.36873   -0.18023  -0.033278    0.12709   -0.36488
   0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278    0.12709    0.36488
   0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.51589
   0.10857   -0.36873    0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   -0.18566    0.21847   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
  -0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857    0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.23433   0.033278   -0.12709   -0.36488
  -0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
    0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488

fx K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

the minsum has worked. Ok, if you look at every column,

(Refer Slide Time: 08:14)



```
K>> treg

treg =

  Columns 1 through 10

   -0.30585  0.0092629   0.040455  0.0078869  -0.017447   0.049335  -0.027401   -0.19991      -0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991    0.22577   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447    0.26568  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629  -0.040455 -0.0078869   0.017447  -0.049335   0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335   0.027401    0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447  -0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   -0.28064
   -0.30585  0.0092629   0.040455   -0.08828   -0.16194   0.049335   0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585 -0.0092629  -0.040455 -0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
    0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991      -0.17   0.029843
   -0.42474  -0.041427  -0.052048  0.0078869   0.017447   0.049335  -0.027401   -0.30923      -0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629  -0.040455  0.0078869   0.017447   0.049335   0.052004   -0.19991      -0.17  -0.029843
   -0.30585  0.0092629   0.040455 -0.0078869   0.017447   0.049335  -0.027401   -0.19991       0.17   0.029843
   -0.30585  0.0092629   0.040455  0.0078869   0.017447   0.049335   0.027401   -0.19991       0.17   0.029843

  Columns 11 through 16

   -0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
    0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   -0.10857    0.42211   -0.18023   0.033278   -0.12709   -0.36488
```

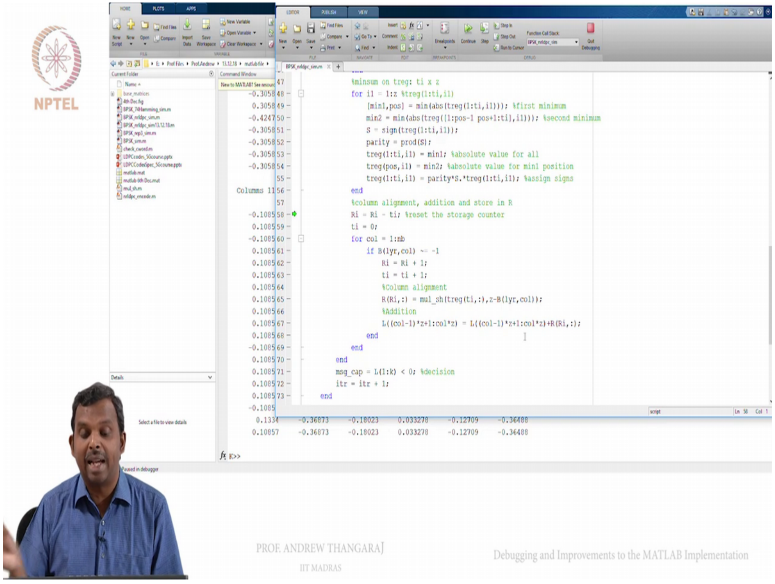
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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)



```
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = 1:nb
            if B(lyr,col) ~= -1
                Ri = Ri + 1;
                ti = ti + 1;
                %Column alignment
                R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                %Addition
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
            end
        end
    end
    msg_cap = L(1:k) < 0; %decision
    itr = itr + 1;
end
```

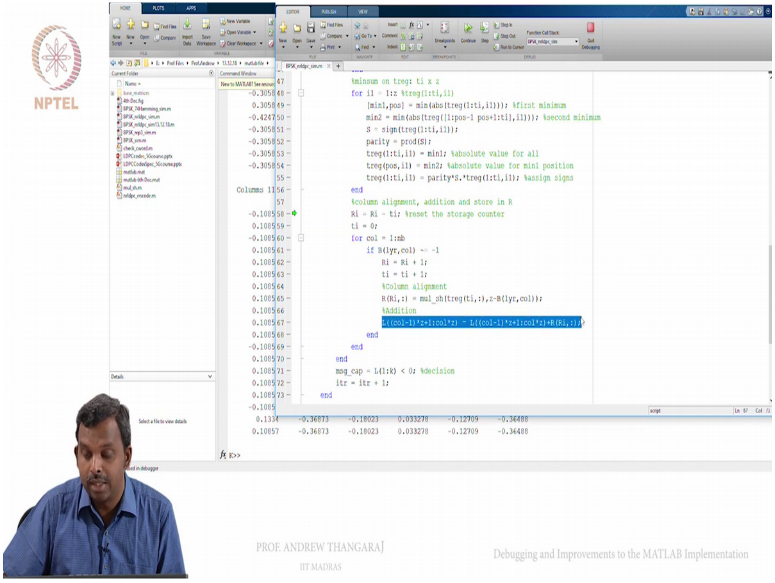
```
    0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488

K>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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)



```
%minsum on treg: ti x z
for i1 = 1:z %treg(1:ti,i1)
    [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
    min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
    S = sign(treg(1:ti,i1));
    parity = prod(S);
    treg(1:ti,i1) = min1; %absolute value for all
    treg(pos,i1) = min2; %absolute value for min1 position
    treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
end
%column alignment, addition and store in R
Ri = Ri - ti; %reset the storage counter
ti = 0;
for col = 1:nb
    if B(lyr,col) ~= -1
        Ri = Ri + 1;
        ti = ti + 1;
        %Column alignment
        R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
        %Addition
        L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
    end
end
```

```
end
msg_cap = L(1:k) < 0; %decision
itr = itr + 1;
end
```

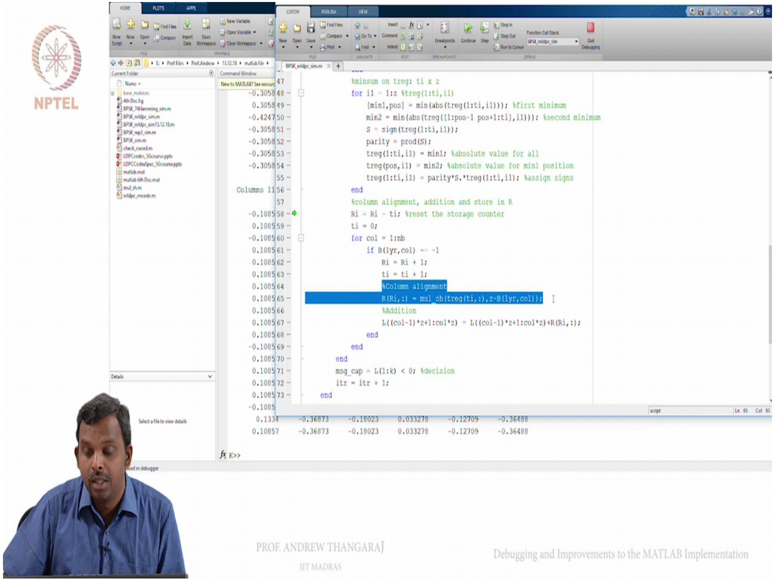
```
0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
0.10857  -0.36873   -0.18023   0.033278   -0.12709   -0.36488
```

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)



```
%minsum on treg: ti x z
for i1 = 1:z %treg(1:ti,i1)
    [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
    min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
    S = sign(treg(1:ti,i1));
    parity = prod(S);
    treg(1:ti,i1) = min1; %absolute value for all
    treg(pos,i1) = min2; %absolute value for min1 position
    treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
end
%column alignment, addition and store in R
Ri = Ri - ti; %reset the storage counter
ti = 0;
for col = 1:nb
    if B(lyr,col) ~= -1
        Ri = Ri + 1;
        ti = ti + 1;
        %Column alignment
        R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
        %Addition
        L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
    end
end
```

```
end
msg_cap = L(1:k) < 0; %decision
itr = itr + 1;
end
```

```
0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
0.10857  -0.36873   -0.18023   0.033278   -0.12709   -0.36488
```

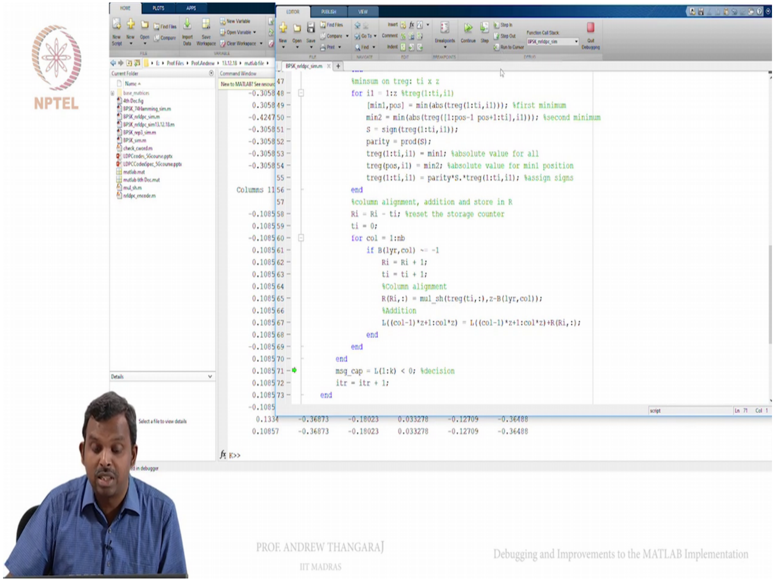
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = 1:nb
            if B(lyr,col) ~= -1
                Ri = Ri + 1;
                ti = ti + 1;
                %Column alignment
                R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                %Addition
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
            end
        end
    end
    msg_cap = L(1:k) < 0; %decision
    itr = itr + 1;
end
```

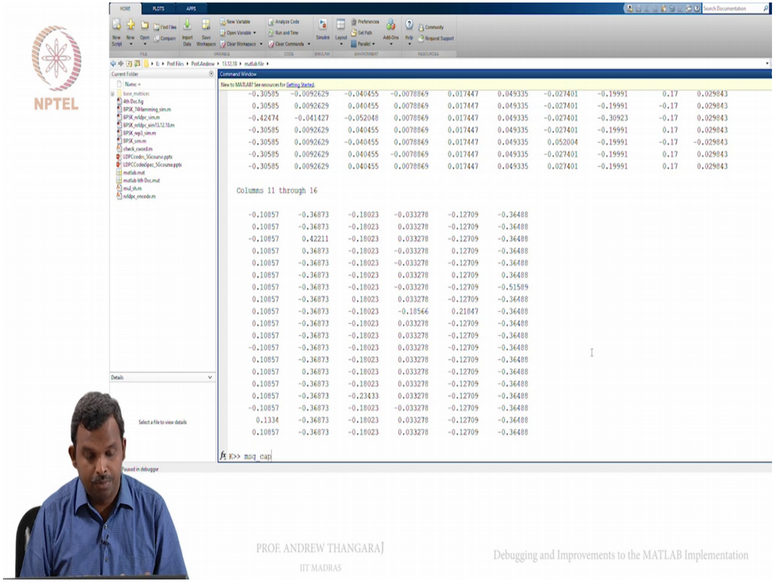
```
   0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
  0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488

K>>
```

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)



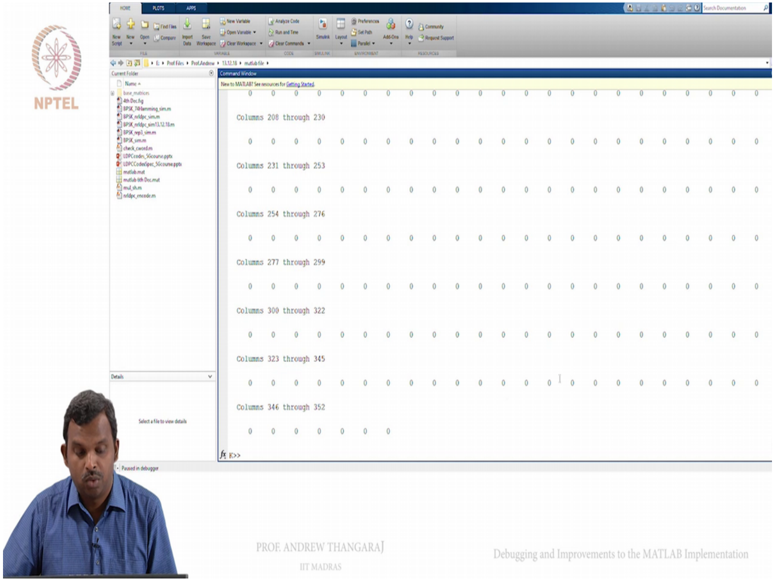
```
  -0.30585  -0.0092629  -0.040455  -0.0078869  0.017447  0.049335  -0.027401  -0.19991   0.17   0.029843
   0.30585   0.0092629   0.040455   0.0078869  0.017447  0.049335  -0.027401  -0.19991  -0.17   0.029843
  -0.42474   -0.041427  -0.052048   0.0078869  0.017447  0.049335  -0.027401  -0.30923  -0.17   0.029843
  -0.30585   0.0092629   0.040455   0.0078869  0.017447  0.049335  -0.027401  -0.19991   0.17   0.029843
  -0.30585   0.0092629  -0.040455   0.0078869  0.017447  0.049335   0.052004  -0.19991  -0.17  -0.029843
  -0.30585   0.0092629   0.040455  -0.0078869  0.017447  0.049335  -0.027401  -0.19991   0.17   0.029843
  -0.30585   0.0092629   0.040455   0.0078869  0.017447  0.049335   0.027401  -0.19991   0.17   0.029843

 Columns 11 through 16

  -0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
  -0.10857    0.42211   -0.18023   0.033278   -0.12709   -0.36488
   0.10857    0.36873   -0.18023  -0.033278    0.12709   -0.36488
   0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278    0.12709    0.36488
   0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.51589
   0.10857   -0.36873    0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   -0.18566    0.21847   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
  -0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857    0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.23433   0.033278   -0.12709   -0.36488
  -0.10857   -0.36873   -0.18023  -0.033278   -0.12709   -0.36488
    0.1334   -0.36873   -0.18023   0.033278   -0.12709   -0.36488
   0.10857   -0.36873   -0.18023   0.033278   -0.12709   -0.36488

K>> msg_cap
```

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: 09:12)

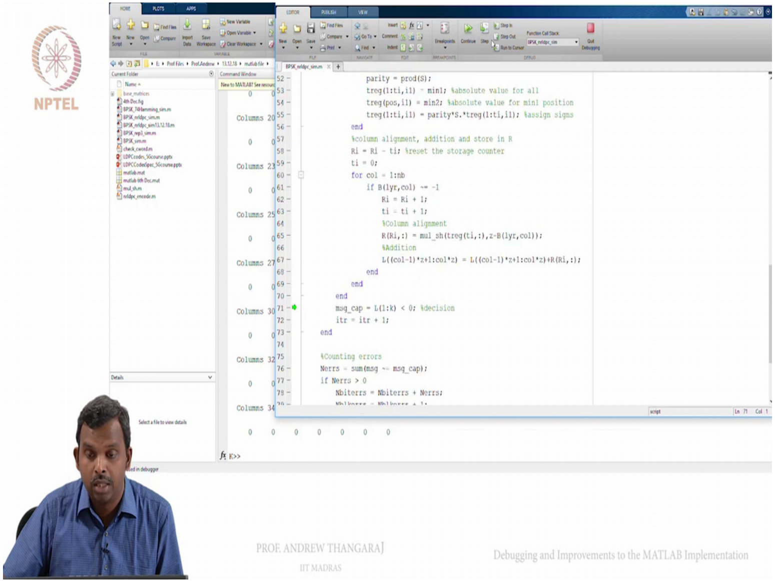


```
  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 208 through 230

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 231 through 253

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 254 through 276

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 277 through 299

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 300 through 322

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 323 through 345

  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Columns 346 through 352

  0     0     0     0     0     0     0

K>>
```

Ok.

So we will count that and then see if there are any errors etc. Ok. So you can let it run through. I have done this program for like 100 blocks, Ok so it will run pretty fast. You can run this and you will get the answer, Ok. So this is 4, 4 d B E b by N naught

(Refer Slide Time: 09:32)

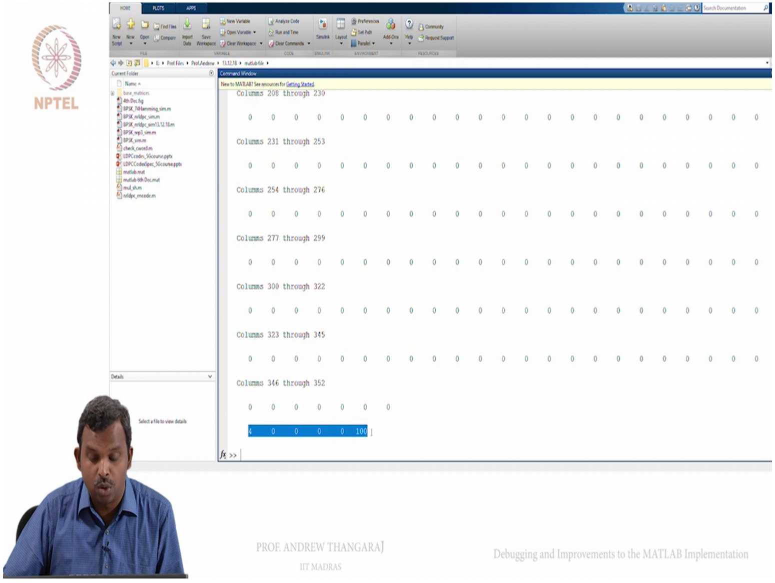


```
                parity = prod(S);
                treg(1:ti,i1) = min1; %absolute value for all
                treg(pos,i1) = min2; %absolute value for min1 position
                treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
            end
            %column alignment, addition and store in R
            Ri = Ri - ti; %reset the storage counter
            ti = 0;
            for col = 1:nb
                if B(lyr,col) ~= -1
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
                end
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
    end

    %Counting errors
    Nerrs = sum(msg ~= msg_cap);
    if Nerrs > 0
        Nbiterrs = Nbiterrs + Nerrs;
        Nblkerrs = Nblkerrs + 1;
```

and let us see. We continue.

It stands for a while. It takes some time. Then it gave you the answer. Ok so there you go. It gave all 0, 0,

(Refer Slide Time: 09:46)



```
  Columns 208 through 230

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 231 through 253

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 254 through 276

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 277 through 299

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 300 through 322

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 323 through 345

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 346 through 352

    0     0     0     0     0     0     0

    4     0     0     0     0   100

fx >>
```

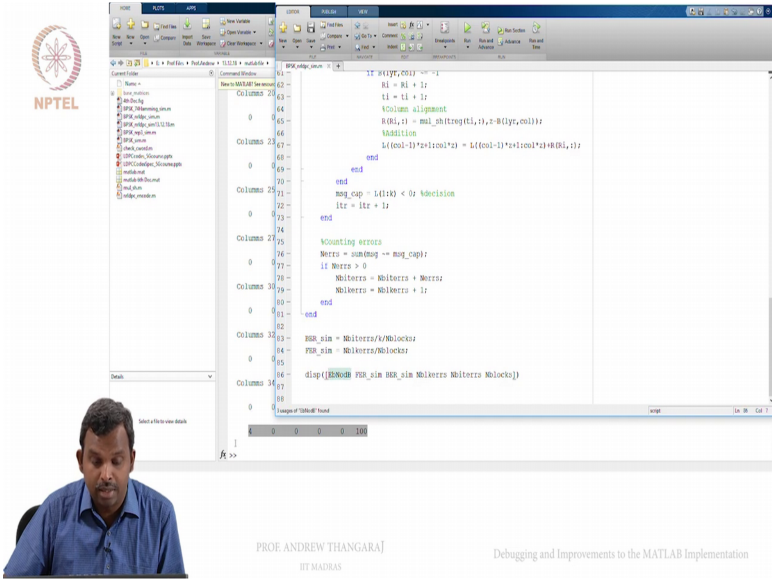
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
                if B(lyr,col) ~= -1
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
                end
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
    end

    %Counting errors
    Nerrs = sum(msg ~= msg_cap);
    if Nerrs > 0
        Nbiterrs = Nbiterrs + Nerrs;
        Nblkerrs = Nblkerrs + 1;
    end
end

BER_sim = Nbiterrs/k/Nblocks;
FER_sim = Nblkerrs/Nblocks;

disp([EbNodB FER_sim BER_sim Nblkerrs Nbiterrs Nblocks])
```

3 usages of "EbNodB" found

```
    4     0     0     0     0   100
```

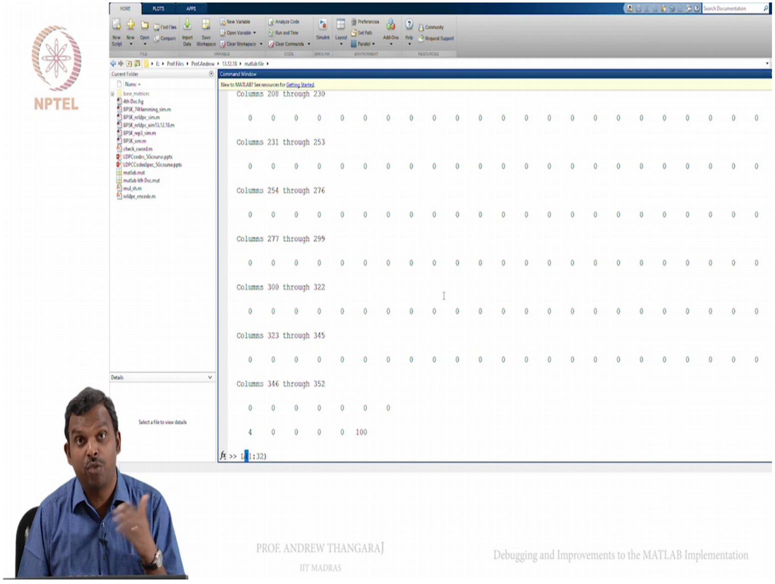
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



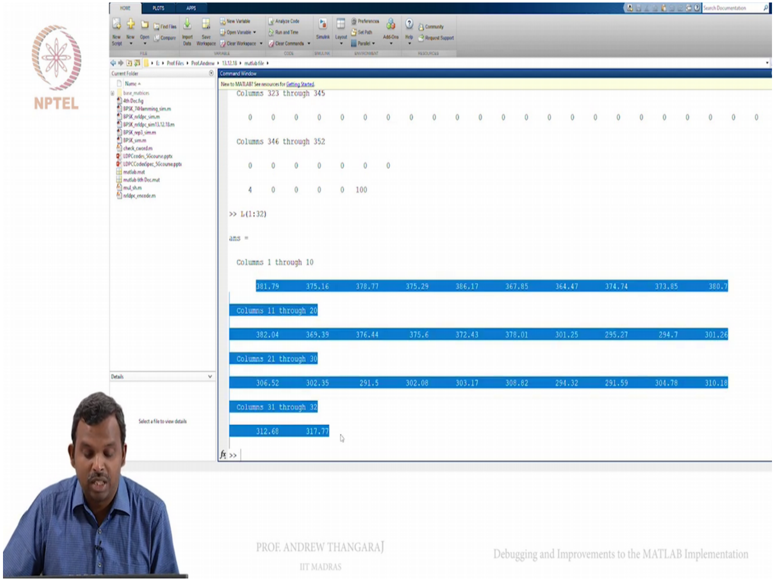
```
  Columns 208 through 230

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 231 through 253

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 254 through 276

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 277 through 299

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 300 through 322

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 323 through 345

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 346 through 352

     0     0     0     0     0     0     0

     4     0     0     0     0   100

>> L(1:32)
```

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)



```
  Columns 323 through 345

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 346 through 352

     0     0     0     0     0     0     0

     4     0     0     0     0   100

>> L(1:32)

ans =

  Columns 1 through 10

    381.79    375.16    378.77    375.29    386.17    367.85    364.47    374.74    373.85     380.7

  Columns 11 through 20

    382.04    369.39    376.44     375.6    372.43    378.01    301.25    295.27     294.7    301.26

  Columns 21 through 30

    306.52    302.35     291.5    302.08    303.17    308.82    294.32    291.59    304.78    310.18

  Columns 31 through 32

    312.68    317.77

>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

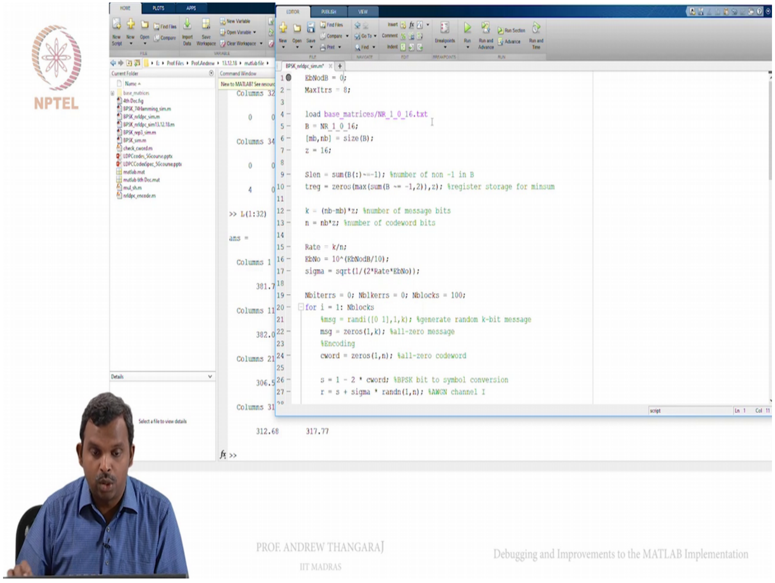
Debugging and Improvements to the MATLAB Implementation

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)

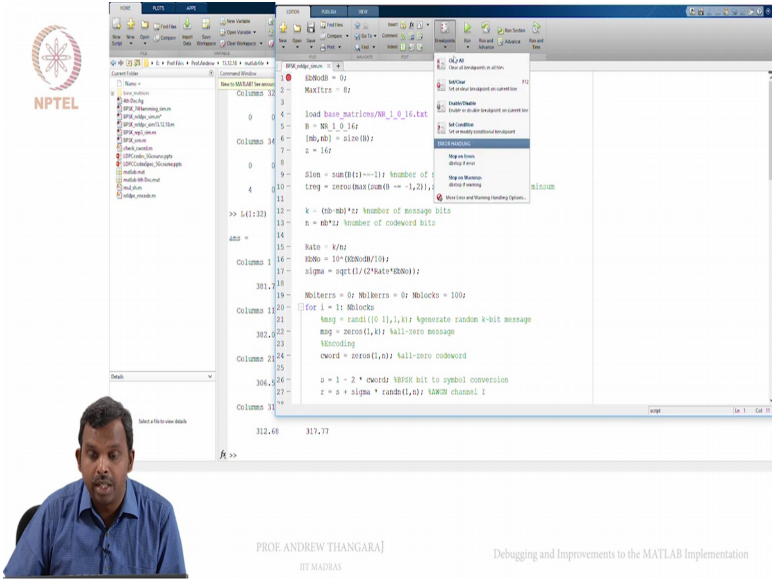


```
EbNodB = 0;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
  Columns 32

    0     0

  Columns 34

    0     0

    4     0

>> L(1:32)

ans =

  Columns 1

     381.7

  Columns 11

     382.0

  Columns 21

     306.5

  Columns 31

     312.68        317.77

fx >>
```

and instead of keeping the

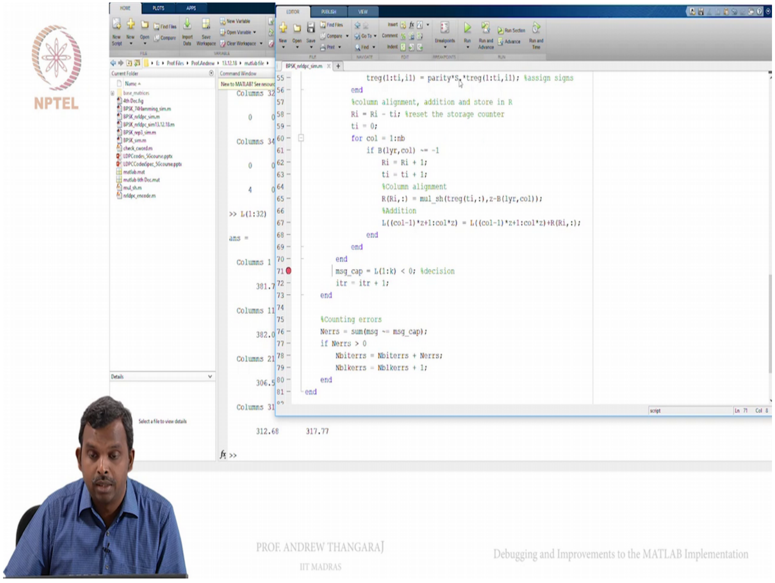
(Refer Slide Time: 10:45)


NPTEL
BPSK_nrldpc_sim.m
Breakpoints
Clear All
Clear all breakpoints in all files
Set/Clear
F12
Set or clear breakpoint on current line
Enable/Disable
Enable or disable breakpoint on current line
Set Condition
Set or modify conditional breakpoint
ERROR HANDLING
Stop on Errors
dbstop if error
Stop on Warnings
dbstop if warning
More Error and Warning Handling Options...
PROF. ANDREW THANGARAJ
IIT MADRAS
Debugging and Improvements to the MATLAB Implementation

```
EbNodB = 0;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of r
treg = zeros(max(sum(B ~= -1,2)),z                                    minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

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

(Refer Slide Time: 10:52)



```
            treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
          end
          %column alignment, addition and store in R
          Ri = Ri - ti; %reset the storage counter
          ti = 0;
          for col = 1:nb
              if B(lyr,col) ~= -1
                  Ri = Ri + 1;
                  ti = ti + 1;
                  %Column alignment
                  R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                  %Addition
                  L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
              end
          end
      end
      msg_cap = L(1:k) < 0; %decision
      itr = itr + 1;
  end

  %Counting errors
  Nerrs = sum(msg ~= msg_cap);
  if Nerrs > 0
      Nbiterrs = Nbiterrs + Nerrs;
      Nblkerrs = Nblkerrs + 1;
  end
end
```

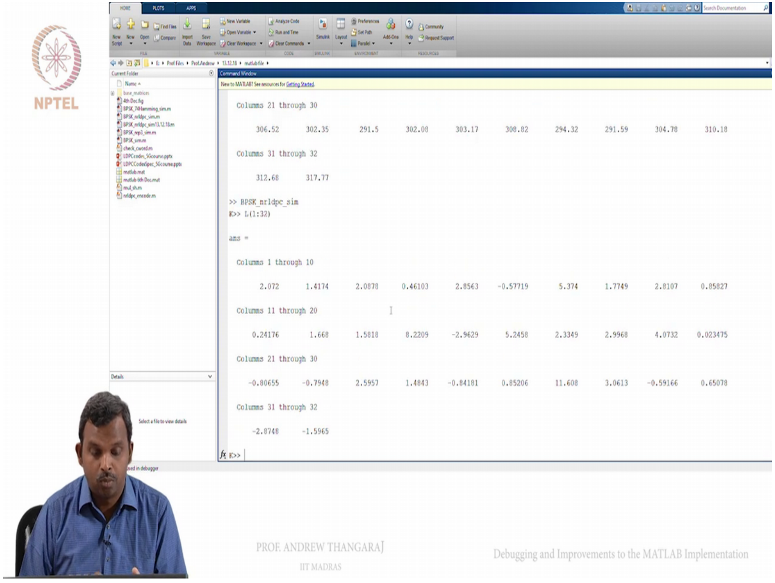
PROF. ANDREW THANGARAJ

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)



```
  Columns 21 through 30

    306.52    302.35     291.5    302.08    303.17    308.82    294.32    291.59    304.78    310.18

  Columns 31 through 32

    312.68    317.77

>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

     2.072    1.4174    2.0878   0.46103    2.8563  -0.57719     5.374    1.7749    2.8107   0.85827

  Columns 11 through 20

   0.24176     1.668    1.5818    8.2209   -2.9629    5.2458    2.3349    2.9968    4.0732  0.023475

  Columns 21 through 30

  -0.80655   -0.7948    2.5957    1.4843  -0.84181   0.85206    11.608    3.0613  -0.59166   0.65078

  Columns 31 through 32

   -2.8748   -1.5965

K>>
```

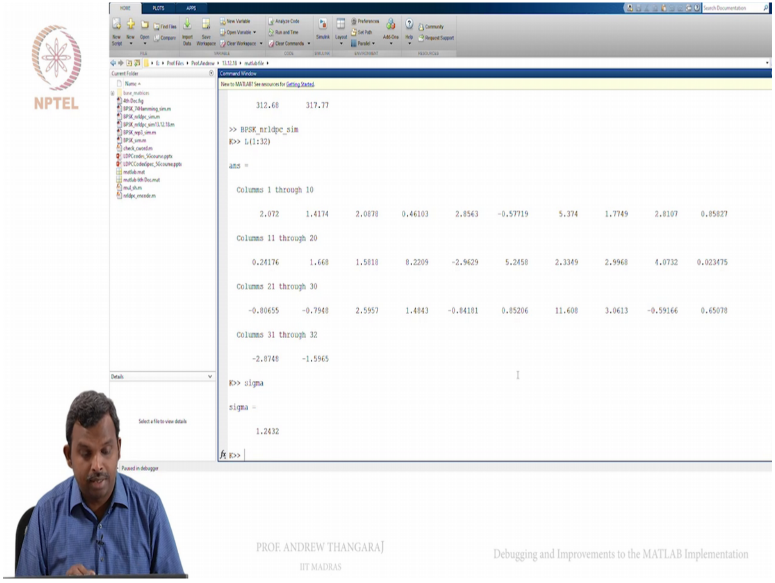
PROF. ANDREW THANGARAJ

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 d B. 0 d B E b over N naught, this

(Refer Slide Time: 11:10)



```
    312.68      317.77

>> BPSK_nrldpc_sim
K>> L(1:32)

ans =

  Columns 1 through 10

     2.072      1.4174      2.0878     0.46103      2.8563    -0.57719       5.374      1.7749      2.8107     0.85827

  Columns 11 through 20

   0.24176       1.668      1.5818      8.2209     -2.9629      5.2458      2.3349      2.9968      4.0732    0.023475

  Columns 21 through 30

  -0.80655     -0.7948      2.5957      1.4843    -0.84181     0.85206      11.608      3.0613    -0.59166     0.65078

  Columns 31 through 32

   -2.8748     -1.5965

K>> sigma

sigma =

    1.2432

K>>
```

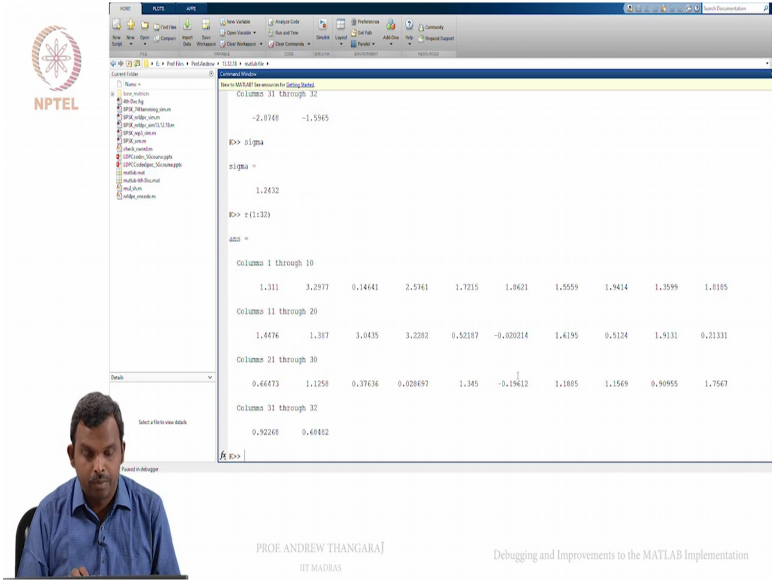
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
  Columns 31 through 32

   -2.8748     -1.5965

K>> sigma

sigma =

    1.2432

K>> r(1:32)

ans =

  Columns 1 through 10

     1.311      3.2977     0.14641      2.5761      1.7215      1.8621      1.5559      1.9414      1.3599      1.8185

  Columns 11 through 20

    1.4476       1.387      3.0435      3.2282     0.52187   -0.020214      1.6195      0.5124      1.9131     0.21331

  Columns 21 through 30

   0.66473      1.1258     0.37636    0.028697       1.345    -0.19612      1.1885      1.1569     0.90955      1.7567

  Columns 31 through 32

   0.92268     0.68482

K>>
```

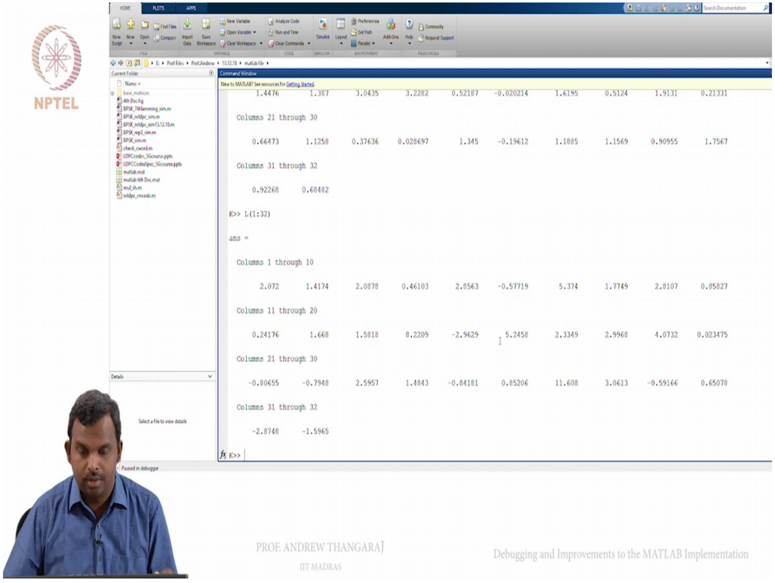
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



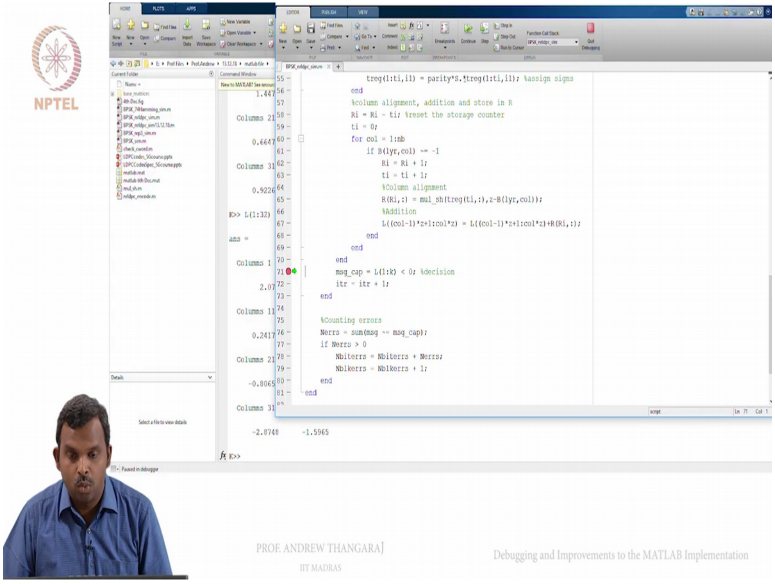
```
     1.4476       1.387      3.0435      3.2282     0.52187   -0.020214      1.6195      0.5124      1.9131     0.21331

  Columns 21 through 30

    0.66473      1.1258     0.37636    0.028697       1.345    -0.19612      1.1885      1.1569     0.90955      1.7567

  Columns 31 through 32

    0.92268     0.68482

K>> L(1:32)

ans =

  Columns 1 through 10

      2.072      1.4174      2.0878     0.46103      2.8563    -0.57719       5.374      1.7749      2.8107     0.85827

  Columns 11 through 20

    0.24176       1.668      1.5818      8.2209     -2.9629      5.2458      2.3349      2.9968      4.0732    0.023475

  Columns 21 through 30

   -0.80655     -0.7948      2.5957      1.4843    -0.84181     0.85206      11.608      3.0613    -0.59166     0.65078

  Columns 31 through 32

    -2.8748     -1.5965

K>>
```

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)



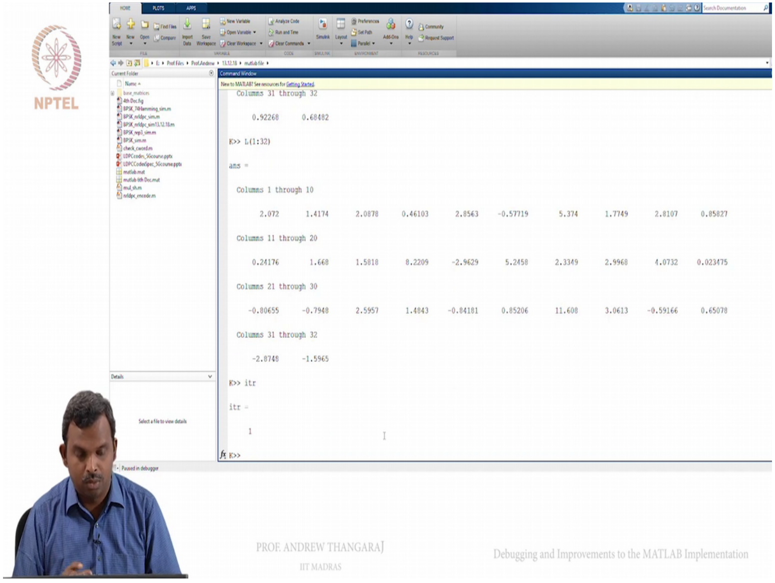
```
                treg(1:ti,il) = parity*S.*treg(1:ti,il); %assign signs
            end
            %column alignment, addition and store in R
            Ri = Ri - ti; %reset the storage counter
            ti = 0;
            for col = 1:nb
                if B(lyr,col) ~= -1
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
                end
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
    end

    %Counting errors
    Nerrs = sum(msg ~= msg_cap);
    if Nerrs > 0
        Nbiterrs = Nbiterrs + Nerrs;
        Nblkerrs = Nblkerrs + 1;
    end
end
```

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    0.68482

K>> L(1:32)

ans =

  Columns 1 through 10

      2.072     1.4174     2.0878    0.46103     2.8563   -0.57719      5.374     1.7749     2.8107    0.85827

  Columns 11 through 20

    0.24176      1.668     1.5818     8.2209    -2.9629     5.2458     2.3349     2.9968     4.0732   0.023475

  Columns 21 through 30

   -0.80655    -0.7948     2.5957     1.4843   -0.84181    0.85206     11.608     3.0613   -0.59166    0.65078

  Columns 31 through 32

    -2.8748    -1.5965

K>> itr

itr =

     1

K>>
```

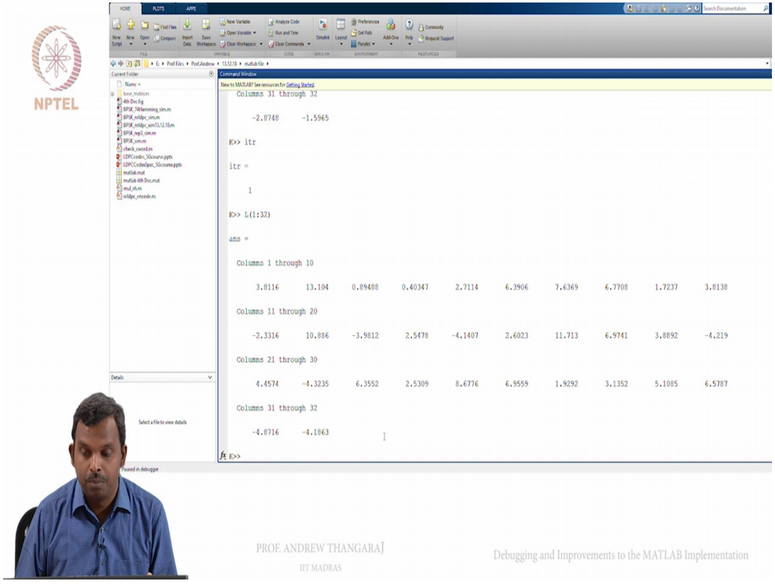
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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    -1.5965

K>> itr

itr =

     1

K>> L(1:32)

ans =

  Columns 1 through 10

     3.8116     13.104    0.89488    0.40347     2.7114     6.3906     7.6369     6.7708     1.7237     3.8138

  Columns 11 through 20

    -2.3316     10.886    -3.9812     2.5478    -4.1407     2.6023     11.713     6.9741     3.8892     -4.219

  Columns 21 through 30

     4.4574    -4.3235     6.3552     2.5309     8.6776     6.9559     1.9292     3.1352     5.1085     6.5787

  Columns 31 through 32

    -4.8716    -4.1863

K>>
```

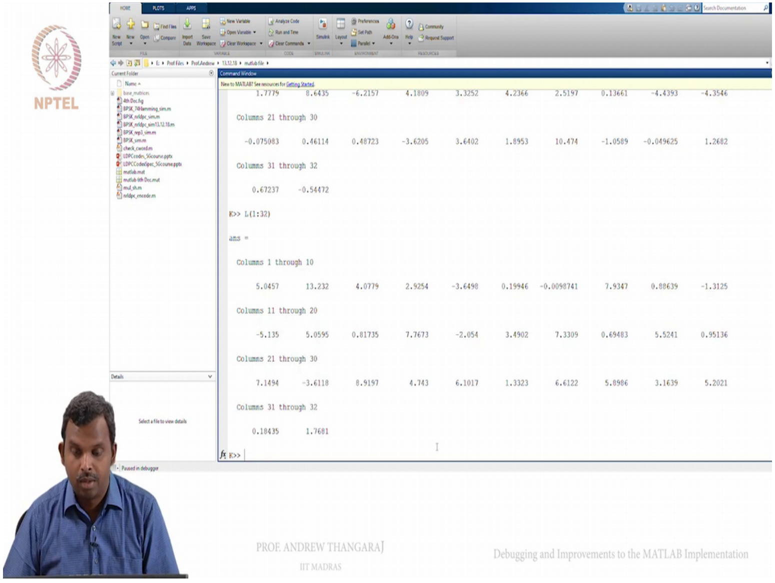
PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

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

Let us continue again. May be 0 d B is too much noise, Ok. Let us see. We are doing 10 iterations. We never know what can happen. Let us continue again.

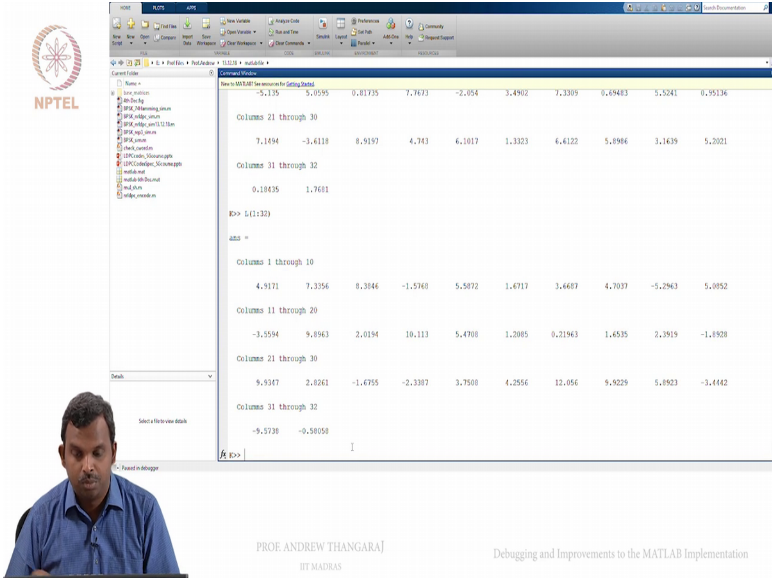
(Refer Slide Time: 12:06)



```
      1.7779      8.6435     -6.2157      4.1809      3.3252      4.2366      2.5197     0.13661     -4.4393     -4.3546

  Columns 21 through 30

   -0.075083     0.46114     0.48723     -3.6205      3.6402      1.8953      10.474     -1.0589   -0.049625      1.2682

  Columns 31 through 32

     0.67237    -0.54472

K>> L(1:32)

ans =

  Columns 1 through 10

      5.0457      13.232      4.0779      2.9254     -3.6498     0.19946  -0.0098741      7.9347     0.88639     -1.3125

  Columns 11 through 20

      -5.135      5.0595     0.81735      7.7673      -2.054      3.4902      7.3309     0.69483      5.5241     0.95136

  Columns 21 through 30

      7.1494     -3.6118      8.9197       4.743      6.1017      1.3323      6.6122      5.8986      3.1639      5.2021

  Columns 31 through 32

     0.18435      1.7681

K>>
```

May be, may be this is some good news. I am not sure. Let us continue again.

(Refer Slide Time: 12:14)

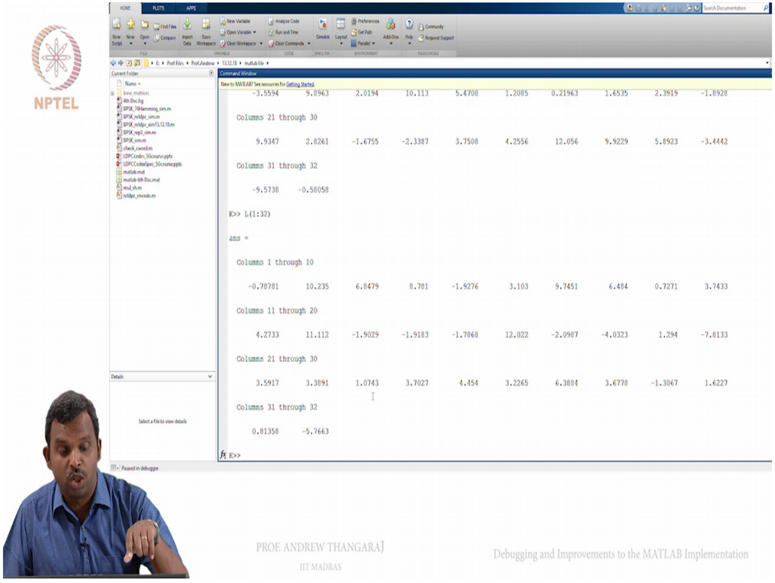


```
      -5.135      5.0595     0.81735      7.7673      -2.054      3.4902      7.3309     0.69483      5.5241     0.95136

  Columns 21 through 30

      7.1494     -3.6118      8.9197       4.743      6.1017      1.3323      6.6122      5.8986      3.1639      5.2021

  Columns 31 through 32

     0.18435      1.7681

K>> L(1:32)

ans =

  Columns 1 through 10

      4.9171      7.3356      8.3846     -1.5768      5.5872      1.6717      3.6687      4.7037     -5.2963      5.0852

  Columns 11 through 20

     -3.5594      9.8963      2.0194      10.113      5.4708      1.2085     0.21963      1.6535      2.3919     -1.8928

  Columns 21 through 30

      9.9347      2.8261     -1.6755     -2.3387      3.7508      4.2556      12.056      9.9229      5.8923     -3.4442

  Columns 31 through 32

     -9.5738    -0.58058

K>>
```

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

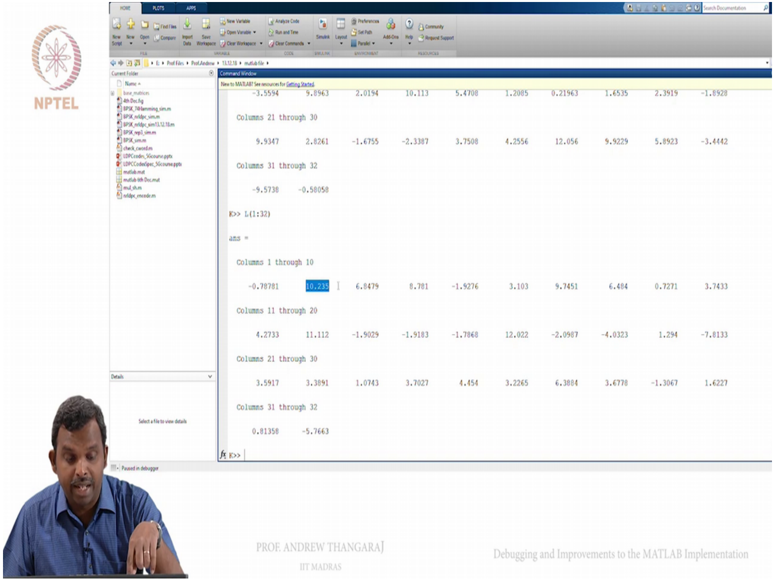
(Refer Slide Time: 12:26)



```
     -3.5594     9.8963     2.0194     10.113     5.4708     1.2085    0.21963     1.6535     2.3919    -1.8928

  Columns 21 through 30

      9.9347     2.8261    -1.6755    -2.3387     3.7508     4.2556     12.056     9.9229     5.8923    -3.4442

  Columns 31 through 32

     -9.5738   -0.58058

K>> L(1:32)

ans =

  Columns 1 through 10

    -0.78781     10.235     6.8479      8.781    -1.9276      3.103     9.7451      6.484     0.7271     3.7433

  Columns 11 through 20

      4.2733     11.112    -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     3.6778    -1.3067     1.6227

  Columns 31 through 32

     0.81358    -5.7663

K>>
```

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

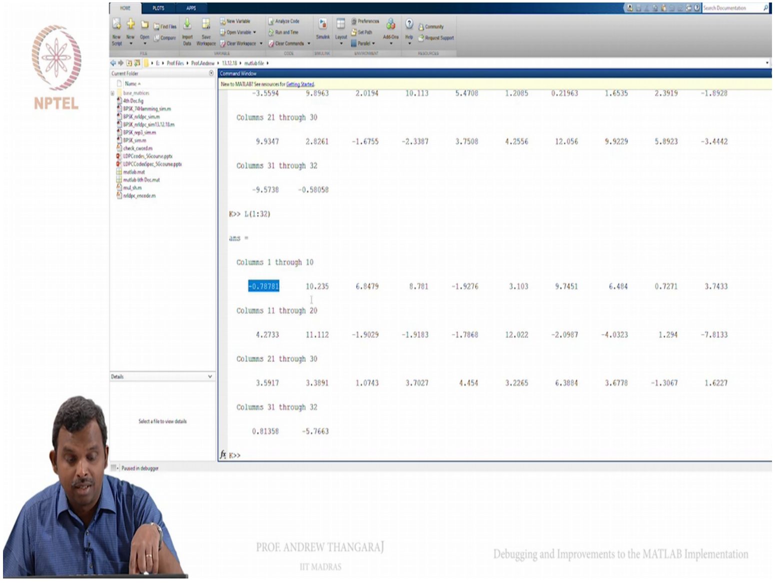
(Refer Slide Time: 12:29)



```
     -3.5594     9.8963     2.0194     10.113     5.4708     1.2085    0.21963     1.6535     2.3919    -1.8928

  Columns 21 through 30

      9.9347     2.8261    -1.6755    -2.3387     3.7508     4.2556     12.056     9.9229     5.8923    -3.4442

  Columns 31 through 32

     -9.5738   -0.58058

K>> L(1:32)

ans =

  Columns 1 through 10

    -0.78781     10.235     6.8479      8.781    -1.9276      3.103     9.7451      6.484     0.7271     3.7433

  Columns 11 through 20

      4.2733     11.112    -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     3.6778    -1.3067     1.6227

  Columns 31 through 32

     0.81358    -5.7663

K>>
```

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

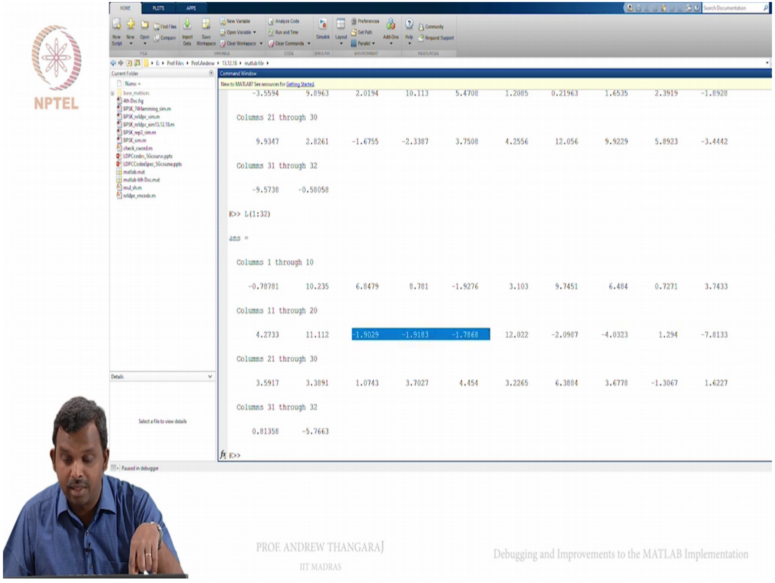
(Refer Slide Time: 12:32)



```
    -3.5594    9.8963    2.0194    10.113    5.4708    1.2085   0.21963    1.6535    2.3919   -1.8928

  Columns 21 through 30

     9.9347    2.8261   -1.6755   -2.3387    3.7508    4.2556    12.056    9.9229    5.8923   -3.4442

  Columns 31 through 32

    -9.5738  -0.58058

K>> L(1:32)

ans =

  Columns 1 through 10

   -0.78781    10.235    6.8479     8.781   -1.9276     3.103    9.7451     6.484    0.7271    3.7433

  Columns 11 through 20

     4.2733    11.112   -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    3.6778   -1.3067    1.6227

  Columns 31 through 32

    0.81358   -5.7663

fx K>>
```

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

(Refer Slide Time: 12:34)

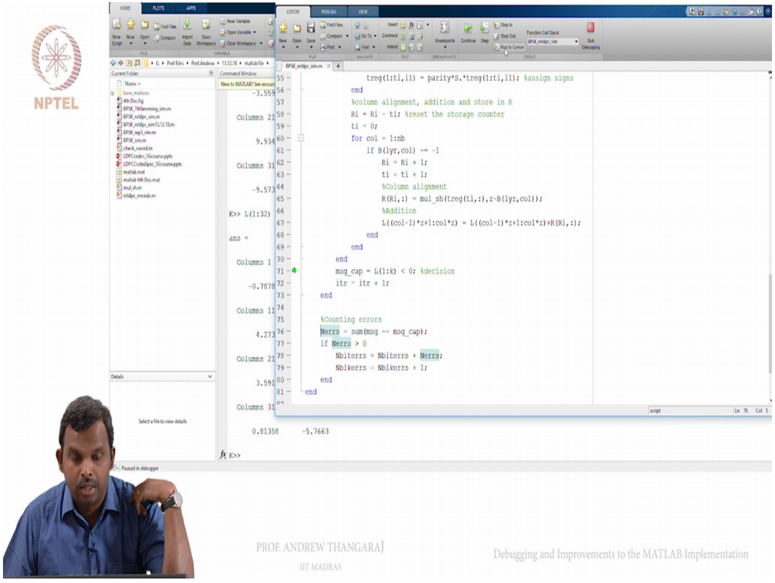


```
    -3.5594    9.8963    2.0194    10.113    5.4708    1.2085   0.21963    1.6535    2.3919   -1.8928

  Columns 21 through 30

     9.9347    2.8261   -1.6755   -2.3387    3.7508    4.2556    12.056    9.9229    5.8923   -3.4442

  Columns 31 through 32

    -9.5738  -0.58058

K>> L(1:32)

ans =

  Columns 1 through 10

   -0.78781    10.235    6.8479     8.781   -1.9276     3.103    9.7451     6.484    0.7271    3.7433

  Columns 11 through 20

     4.2733    11.112   -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    3.6778   -1.3067    1.6227

  Columns 31 through 32

    0.81358   -5.7663

fx K>>
```

Ok. Anything negative is an error, Ok.

So 0 d B E b over N naught, 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

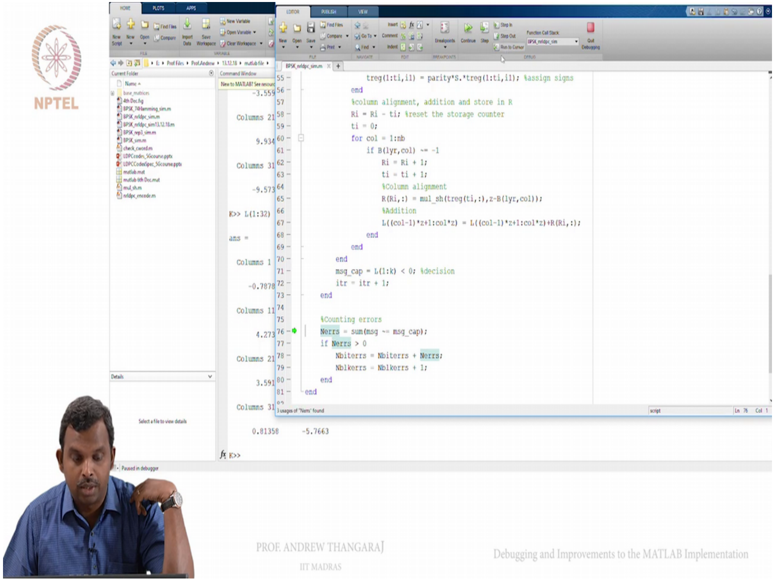
(Refer Slide Time: 12:52)



```
                treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
            end
            %column alignment, addition and store in R
            Ri = Ri - ti; %reset the storage counter
            ti = 0;
            for col = 1:nb
                if B(lyr,col) ~= -1
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
                end
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
    end

    %Counting errors
    Nerrs = sum(msg ~= msg_cap);
    if Nerrs > 0
        Nbiterrs = Nbiterrs + Nerrs;
        Nblkerrs = Nblkerrs + 1;
    end
end
```

and run to cursor, Ok.

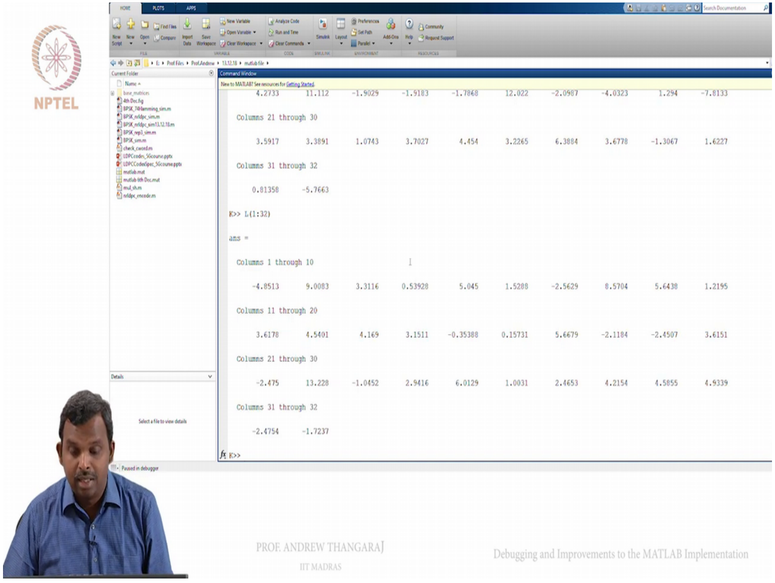
(Refer Slide Time: 12:54)



```
                treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
            end
            %column alignment, addition and store in R
            Ri = Ri - ti; %reset the storage counter
            ti = 0;
            for col = 1:nb
                if B(lyr,col) ~= -1
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
                end
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
    end

    %Counting errors
    Nerrs = sum(msg ~= msg_cap);
    if Nerrs > 0
        Nbiterrs = Nbiterrs + Nerrs;
        Nblkerrs = Nblkerrs + 1;
    end
end
```

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

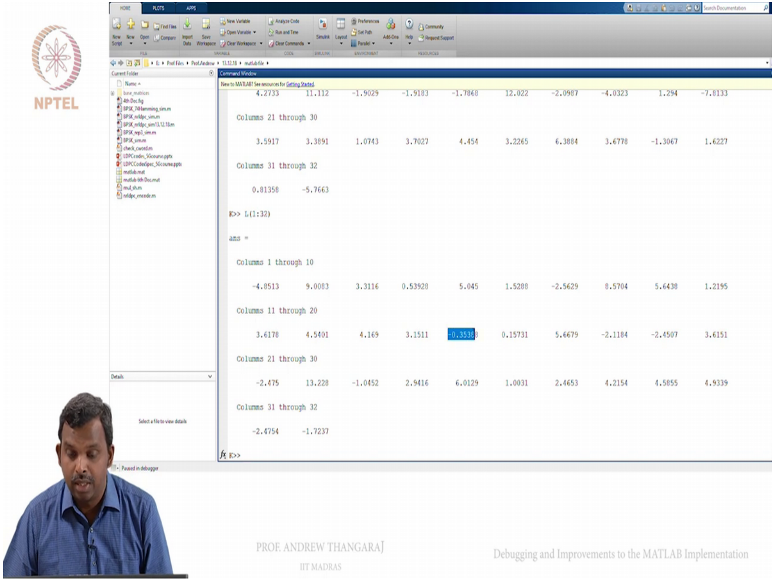
(Refer Slide Time: 12:59)



```
    4.2733     11.112    -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     3.6778    -1.3067     1.6227

  Columns 31 through 32

   0.81358    -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

   -4.8513     9.0083     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    -2.1184    -2.4507     3.6151

  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

fx K>>
```

There is still some negatives.

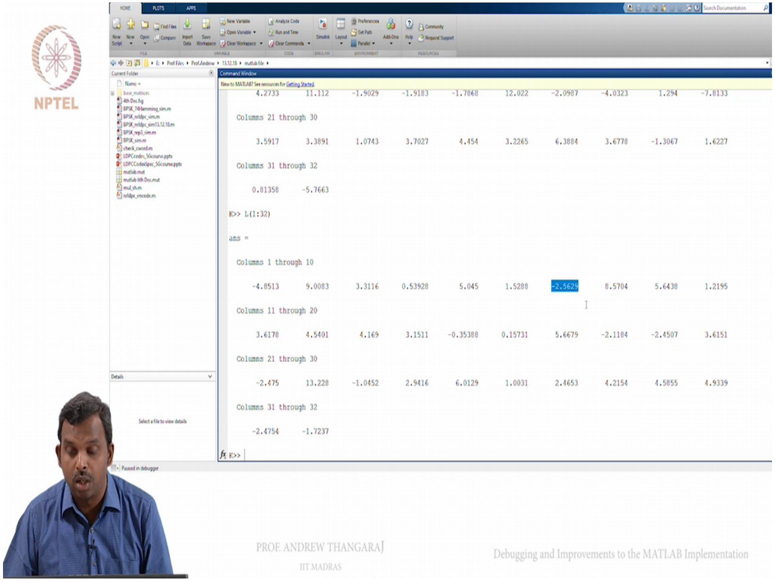
(Refer Slide Time: 13:01)



```
    4.2733     11.112    -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     3.6778    -1.3067     1.6227

  Columns 31 through 32

   0.81358    -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

   -4.8513     9.0083     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    -2.1184    -2.4507     3.6151

  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

fx K>>
```

Here are still negatives so this will be erroneous block,

(Refer Slide Time: 13:03)

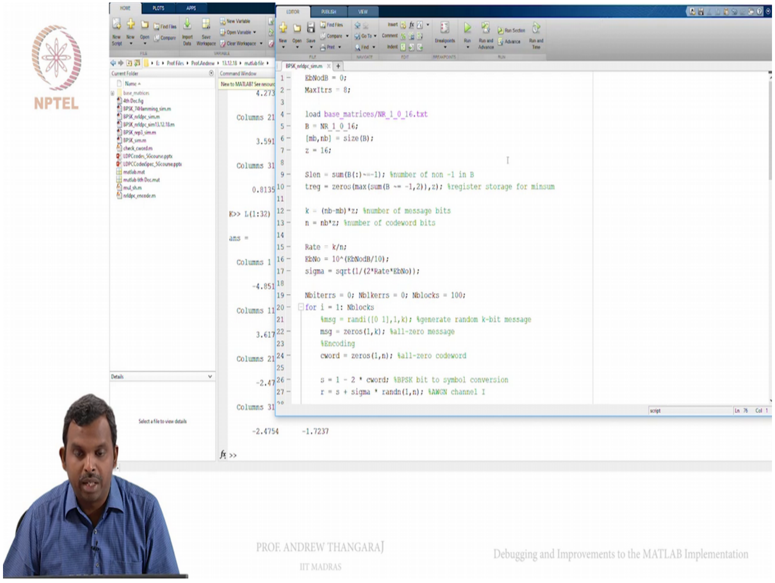


```
      4.2733      11.112     -1.9029     -1.9183     -1.7868      12.022     -2.0987     -4.0323       1.294     -7.8133

  Columns 21 through 30

      3.5917      3.3891      1.0743      3.7027       4.454      3.2265      6.3884      3.6778     -1.3067      1.6227

  Columns 31 through 32

     0.81358     -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

     -4.8513      9.0083      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     -2.1184     -2.4507      3.6151

  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

K>>
```

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)

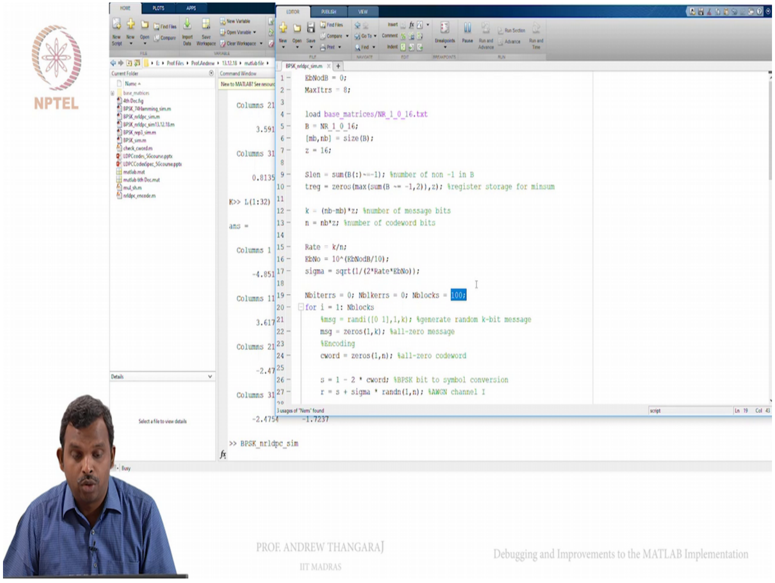


```
EbNodB = 0;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel
```

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

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

(Refer Slide Time: 13:25)

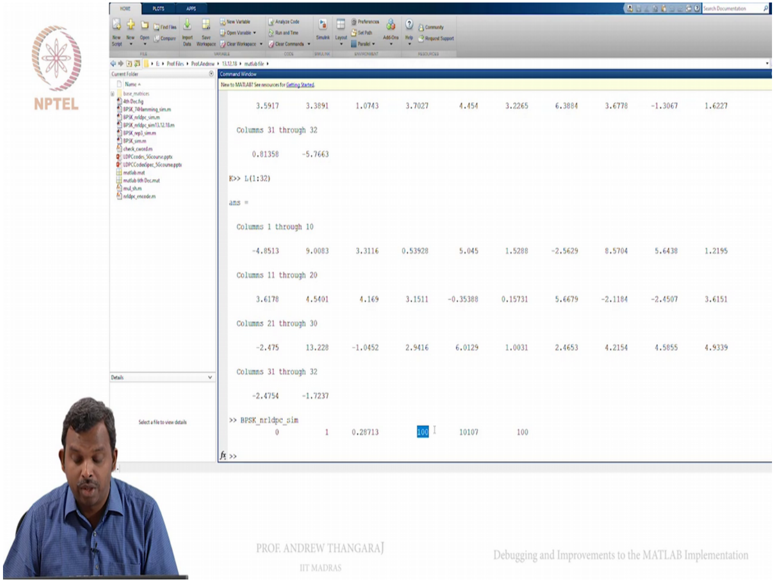


```
EbNodB = 0;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
>> BPSK_nrldpc_sim
```

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

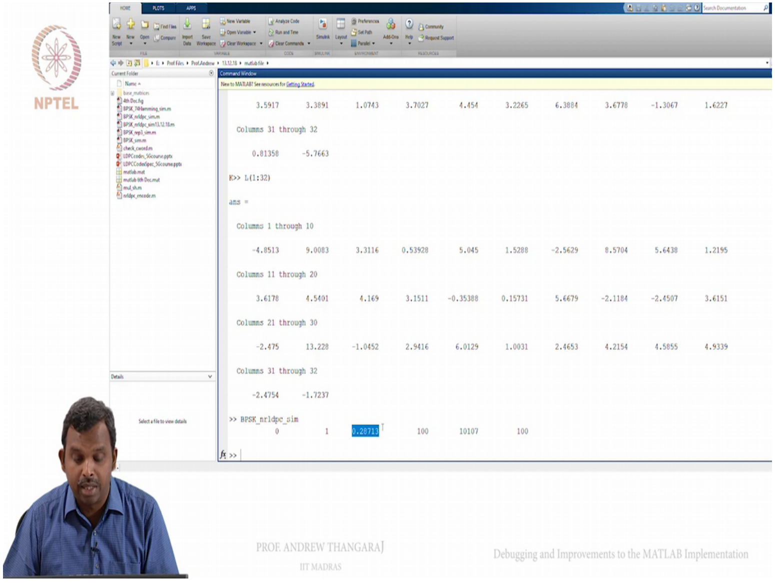
(Refer Slide Time: 13:33)



```
     3.5917      3.3891      1.0743      3.7027       4.454      3.2265      6.3884      3.6778     -1.3067      1.6227

  Columns 31 through 32

    0.81358     -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

    -4.8513      9.0083      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     -2.1184     -2.4507      3.6151

  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_nrldpc_sim
          0           1     0.28713         100       10107         100

>>
```

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

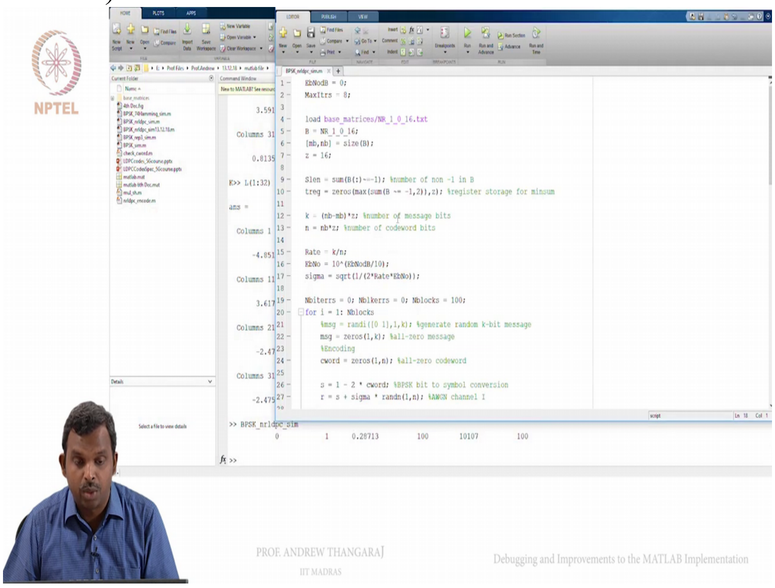
(Refer Slide Time: 13:37)



```
     3.5917      3.3891      1.0743      3.7027       4.454      3.2265      6.3884      3.6778     -1.3067      1.6227

  Columns 31 through 32

    0.81358     -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

    -4.8513      9.0083      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     -2.1184     -2.4507      3.6151

  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_nrldpc_sim
          0           1     0.28713         100       10107         100

fx >>
```

rate was point 2 8, not too bad, Ok. So that is

(Refer Slide Time: 13:40)



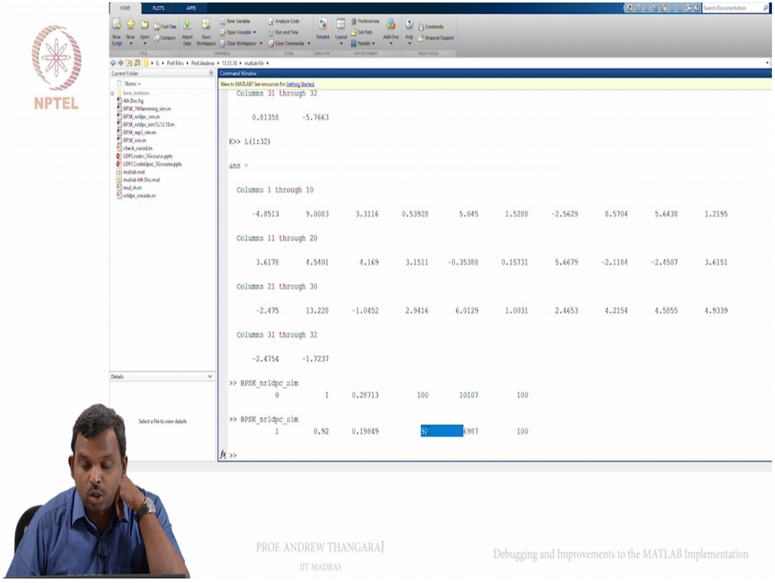
```
EbNodB = 0;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
>> BPSK_nrldpc_sim
          0           1     0.28713         100       10107         100
```

the situation with E b over N naught of 0 d B.

So maybe we will do 1 d B. Let us see what happens, Ok. So this is taking a little time. So you see there is lot of value in writing more efficient code if you want quick answer. So for instance, Ok so anyway. So let me finish the simulation. At 1 d B you are already seeing there are 8

(Refer Slide Time: 14:01)



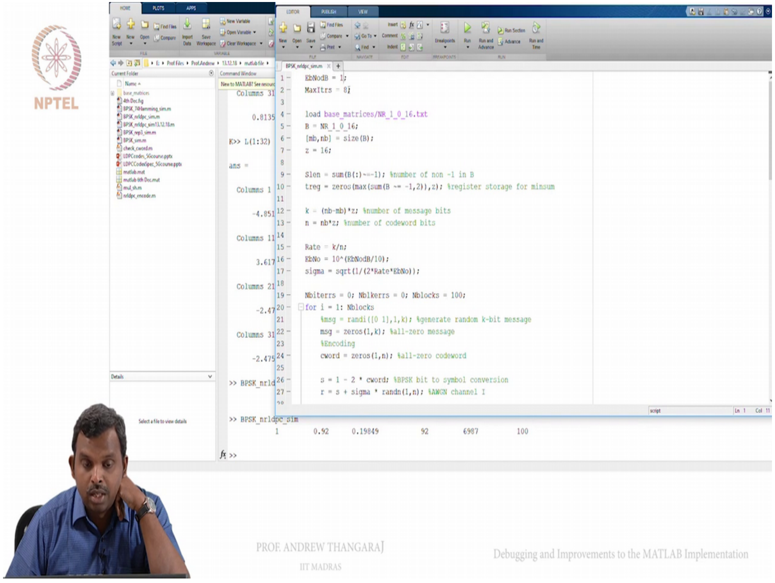
```
  Columns 31 through 32

      0.81358      -5.7663

K>> L(1:32)

ans =

  Columns 1 through 10

     -4.8513       9.0083       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      -2.1184      -2.4507       3.6151

  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_nrldpc_sim
           0            1      0.28713          100        10107          100

>> BPSK_nrldpc_sim
           1         0.92      0.19849           92         6987          100

fx >>
```

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)



```
EbNodB = 1;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
>> BPSK_nrldpc_sim
           1         0.92      0.19849           92         6987          100
```

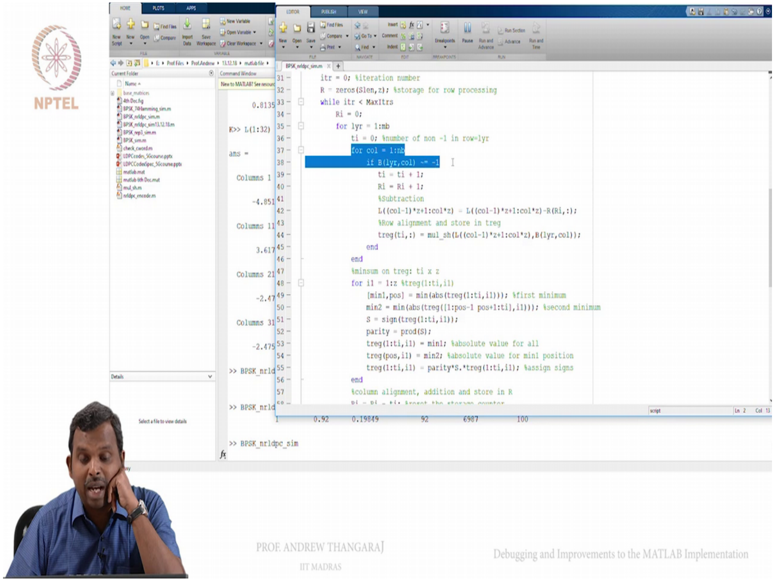
PROF. ANDREW THANGARAJ
IIT MADRAS

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: 14:19)



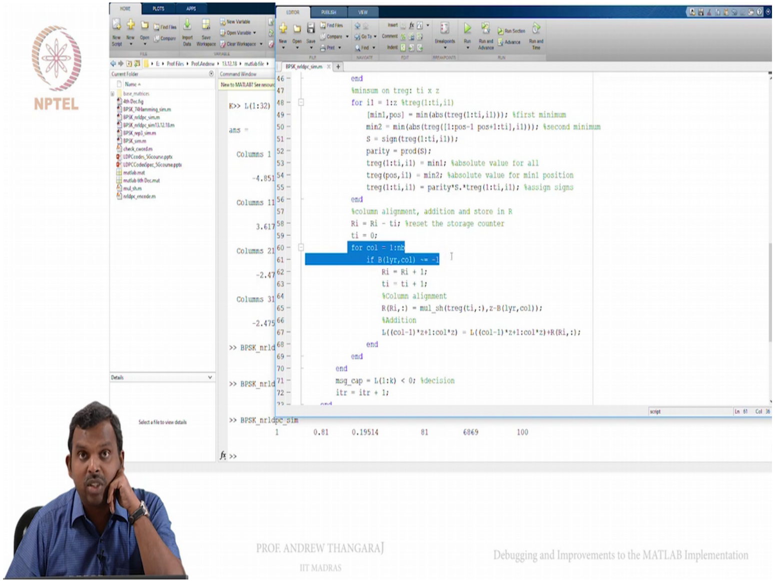
```
itr = 0; %iteration number
R = zeros(Slen,z); %storage for row processing
while itr < MaxItrs
    Ri = 0;
    for lyr = 1:mb
        ti = 0; %number of non -1 in row=lyr
        for col = 1:nb
            if B(lyr,col) ~= -1
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
```

```
1     0.92     0.19849          92        6987          100

>> BPSK_nrldpc_sim
```

badly written, Ok.

So those of you who know MATLAB code, you know that this loop is very, very badly written. I can very easily cut this short significantly, Ok. Ok because the same loop comes even here. So this loop can be skipped

(Refer Slide Time: 14:32)

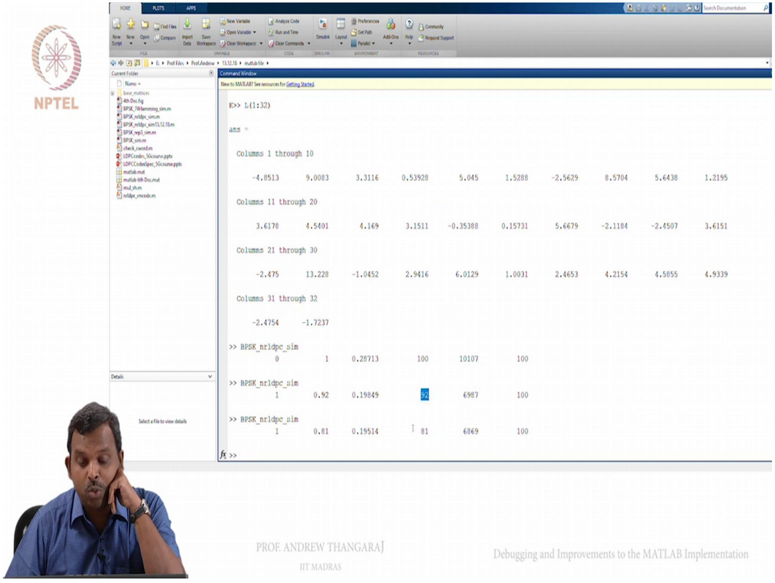


```
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = 1:nb
            if B(lyr,col) ~= -1
                Ri = Ri + 1;
                ti = ti + 1;
                %Column alignment
                R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                %Addition
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
            end
        end
    end
    msg_cap = L(1:k) < 0; %decision
    itr = itr + 1;
```

```
>> BPSK_nrldpc_sim
1     0.81     0.19514          81        6869          100
```

It speeded up in the next time we run it. But you see here, when you did 16 iterations already there were some

(Refer Slide Time: 14:37)



```
K>> L(1:32)

ans =

  Columns 1 through 10

    -4.8513    9.0083    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   -2.1184   -2.4507    3.6151

  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_nrldpc_sim
         0         1   0.28713       100     10107       100

>> BPSK_nrldpc_sim
         1      0.92   0.19849        92      6987       100

>> BPSK_nrldpc_sim
         1      0.81   0.19514        81      6869       100

>>
```

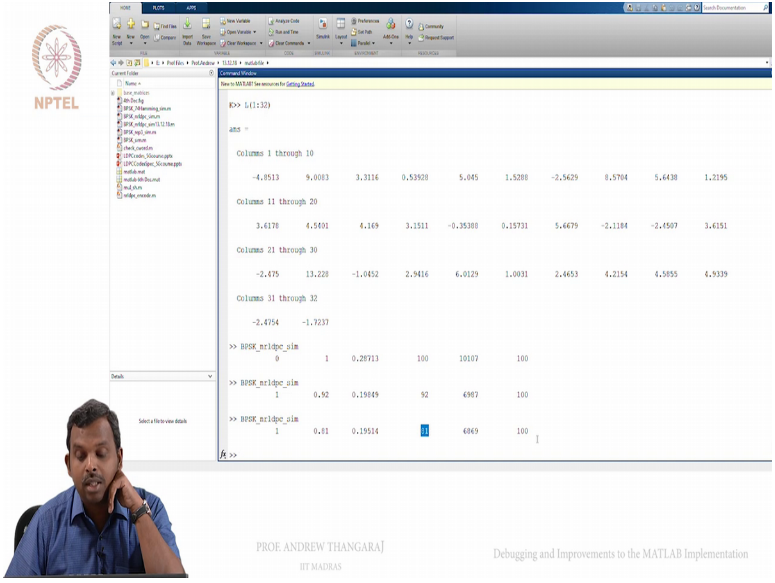
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
K>> L(1:32)

ans =

  Columns 1 through 10

    -4.8513    9.0083    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   -2.1184   -2.4507    3.6151

  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_nrldpc_sim
         0         1   0.28713       100     10107       100

>> BPSK_nrldpc_sim
         1      0.92   0.19849        92      6987       100

>> BPSK_nrldpc_sim
         1      0.81   0.19514        81      6869       100

>>
```

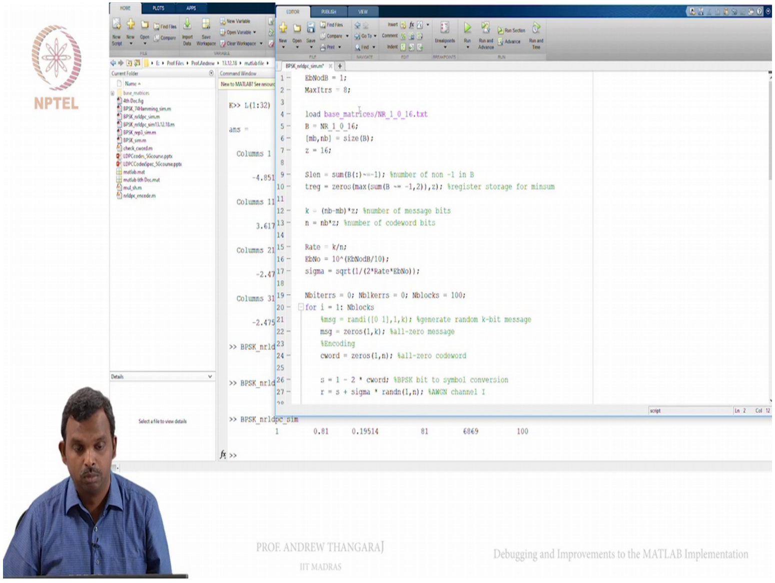
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
EbNodB = 1;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

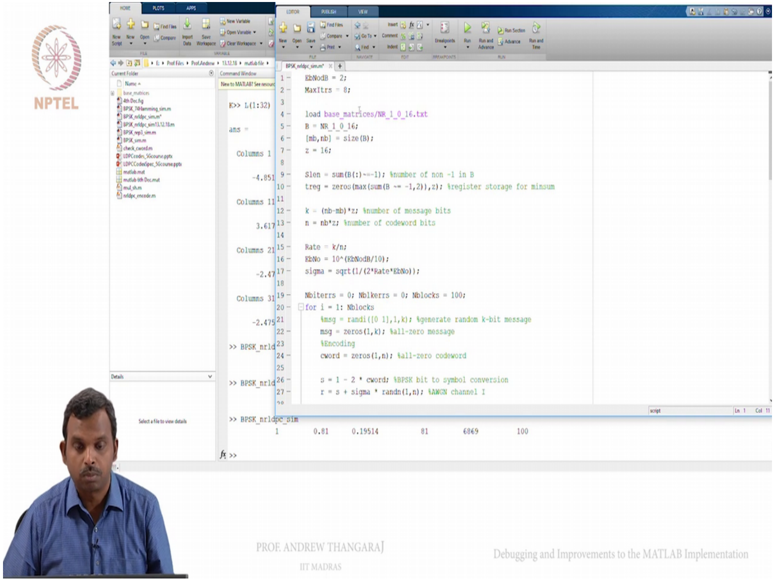
```
K>> L(1:32)

ans =

  Columns 1

    -4.851

  Columns 11

     3.617

  Columns 21

     -2.47

  Columns 31

    -2.475

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrldpc_sim
         1       0.81     0.19514          81         6869          100

fx >>
```

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)



```
EbNodB = 2;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_16;
[mb,nb] = size(B);
z = 16;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
K>> L(1:32)

ans =

  Columns 1

    -4.851

  Columns 11

     3.617

  Columns 21

     -2.47

  Columns 31

    -2.475

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrldpc_sim
         1       0.81     0.19514          81         6869          100

fx >>
```

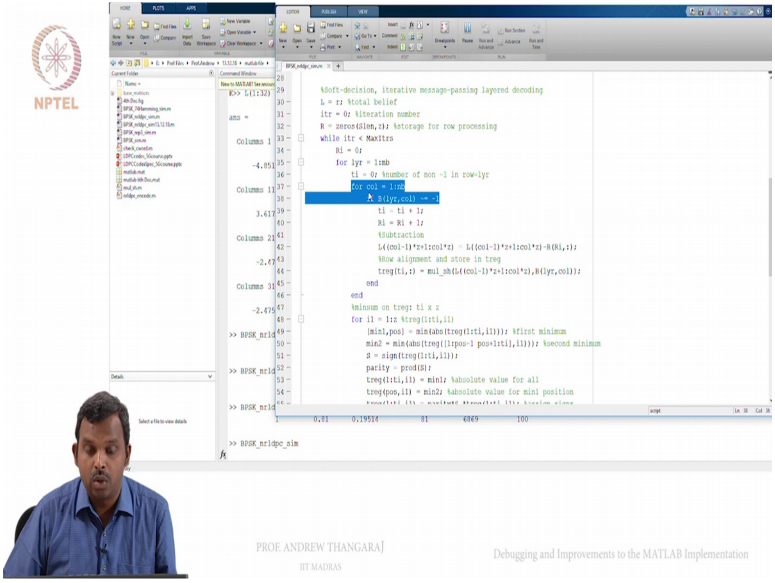
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



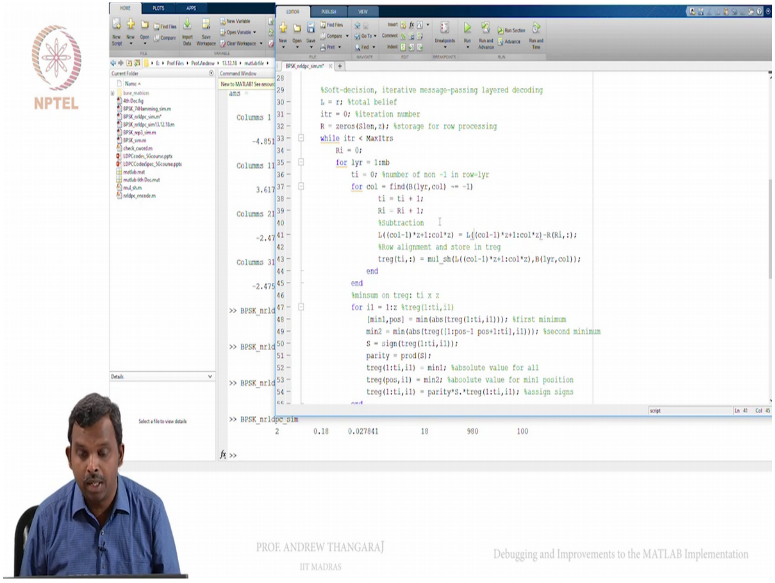
```
%Soft-decision, iterative message-passing layered decoding
L = r; %total belief
itr = 0; %iteration number
R = zeros(Slen,z); %storage for row processing
while itr < MaxItrs
    Ri = 0;
    for lyr = 1:mb
        ti = 0; %number of non -1 in row-lyr
        for col = 1:nb
            if B(lyr,col) ~= -1
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
```

```
1      0.81     0.19514          81          6869          100

>> BPSK_nrldpc_sim
```

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)

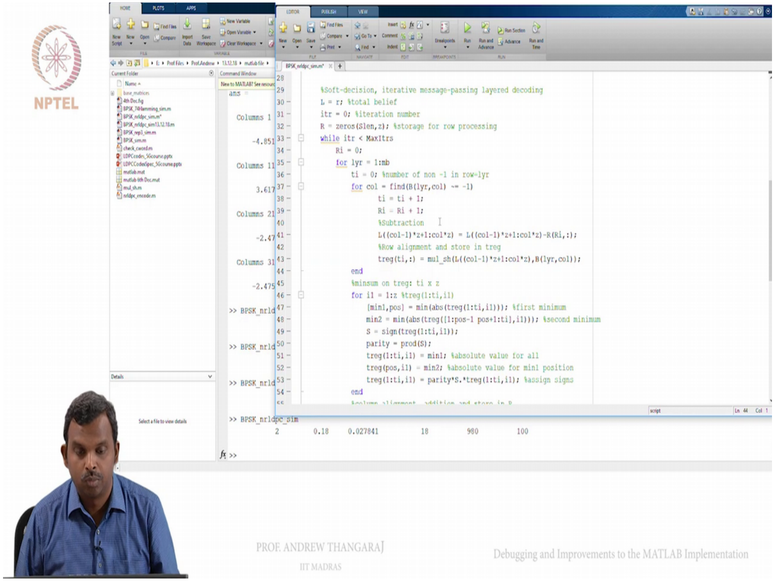


```
%Soft-decision, iterative message-passing layered decoding
L = r; %total belief
itr = 0; %iteration number
R = zeros(Slen,z); %storage for row processing
while itr < MaxItrs
    Ri = 0;
    for lyr = 1:mb
        ti = 0; %number of non -1 in row-lyr
        for col = find(B(lyr,col) ~= -1)
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
```

```
>> BPSK_nrldpc_sim
2      0.18     0.027841          18          980          100
```

and then you got rid of one loop,

(Refer Slide Time: 15:46)



```
%Soft-decision, iterative message-passing layered decoding
L = r; %total belief
itr = 0; %iteration number
R = zeros(Slen,z); %storage for row processing
while itr < MaxItrs
    Ri = 0;
    for lyr = 1:mb
        ti = 0; %number of non -1 in row-lyr
        for col = find(B(lyr,col) ~= -1)
                ti = ti + 1;
                Ri = Ri + 1;
                %Subtraction
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
```

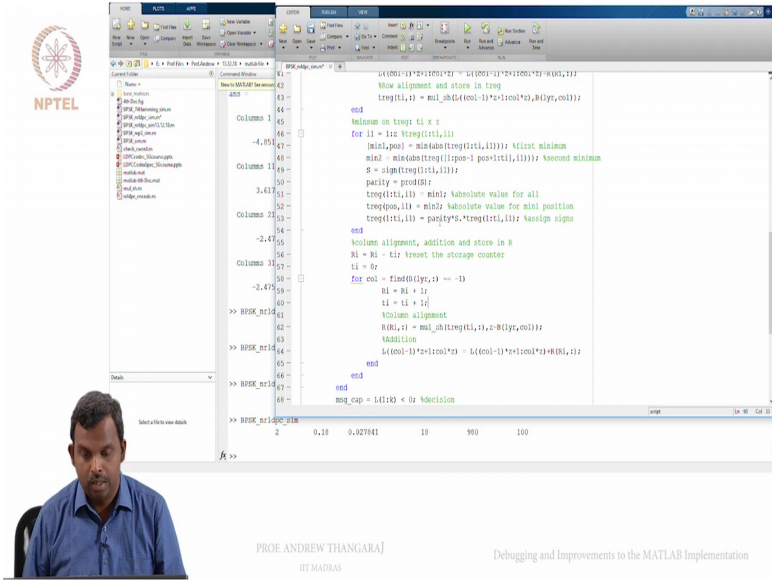
```
>> BPSK_nrldpc_sim
     2     0.18     0.027841     18     980     100
```

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)

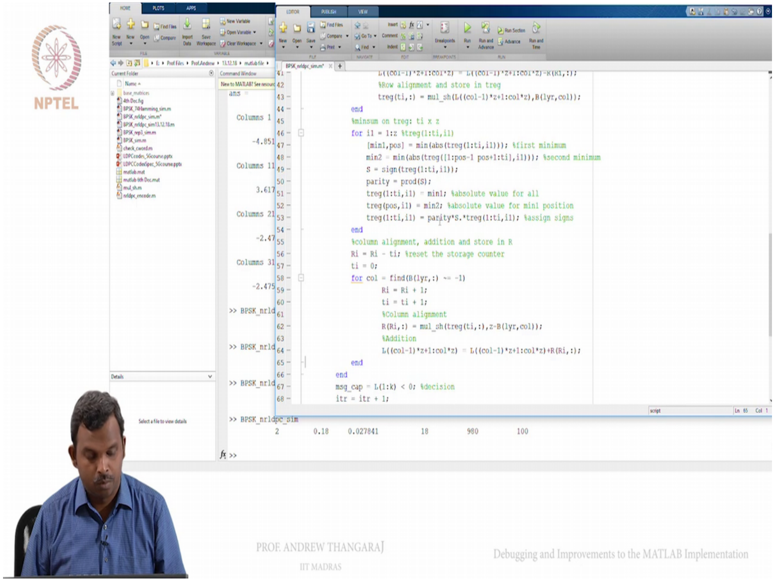


```
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
        end
        %minsum on treg: ti x z
        for i1 = 1:z %treg(1:ti,i1)
            [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
            min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
            S = sign(treg(1:ti,i1));
            parity = prod(S);
            treg(1:ti,i1) = min1; %absolute value for all
            treg(pos,i1) = min2; %absolute value for min1 position
            treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
        end
        %column alignment, addition and store in R
        Ri = Ri - ti; %reset the storage counter
        ti = 0;
        for col = find(B(lyr,:) ~= -1)
                Ri = Ri + 1;
                ti = ti + 1;
                %Column alignment
                R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                %Addition
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
            end
        end
    end
    msg_cap = L(1:k) < 0; %decision
```

```
>> BPSK_nrldpc_sim
     2     0.18     0.027841     18     980     100
```

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

(Refer Slide Time: 16:19)

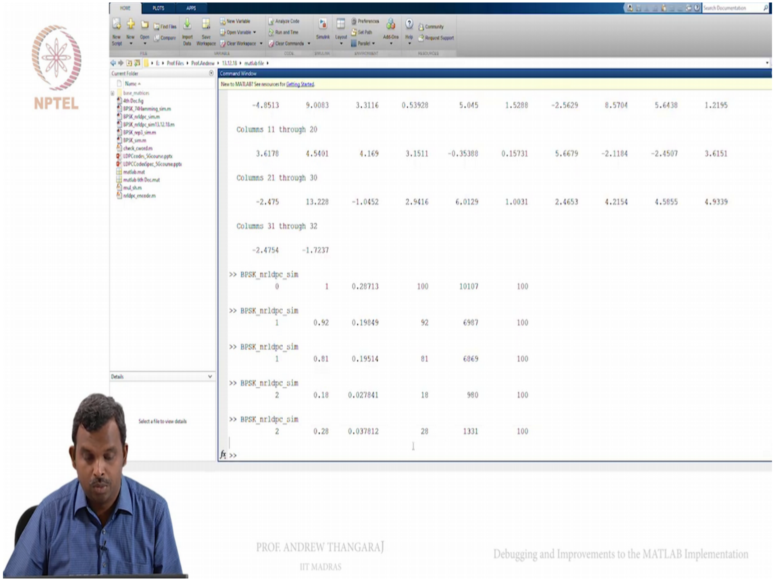


```
                L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                %Row alignment and store in treg
                treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
            end
            %minsum on treg: ti x z
            for i1 = 1:z %treg(1:ti,i1)
                [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
                min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
                S = sign(treg(1:ti,i1));
                parity = prod(S);
                treg(1:ti,i1) = min1; %absolute value for all
                treg(pos,i1) = min2; %absolute value for min1 position
                treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
            end
            %column alignment, addition and store in R
            Ri = Ri - ti; %reset the storage counter
            ti = 0;
            for col = find(B(lyr,:) ~= -1)
                    Ri = Ri + 1;
                    ti = ti + 1;
                    %Column alignment
                    R(Ri,:) = mul_sh(treg(ti,:),z-B(lyr,col));
                    %Addition
                    L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)+R(Ri,:);
            end
        end
        msg_cap = L(1:k) < 0; %decision
        itr = itr + 1;
```

```
>> BPSK_nrldpc_sim
        2        0.18      0.027841        18        980        100
```

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)



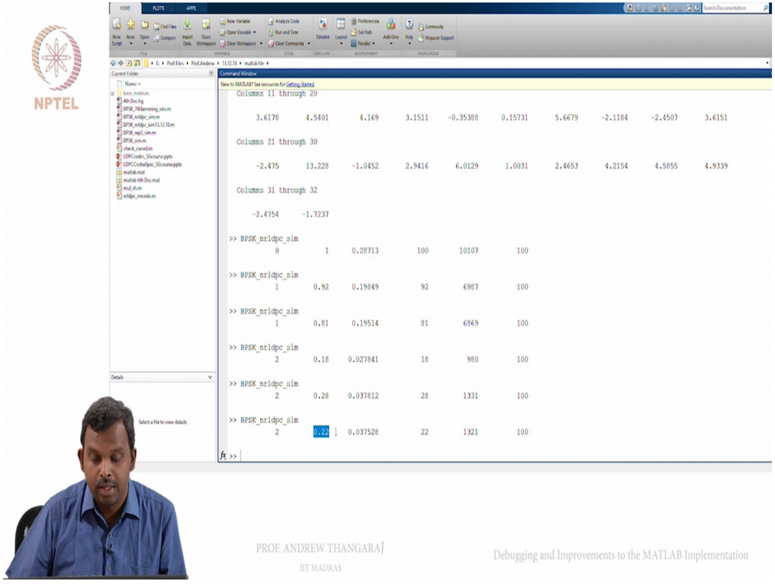
```
   -4.8513    9.0083    3.3116   0.53928     5.045    1.5288   -2.5629    8.5704    5.6438    1.2195

  Columns 11 through 20

    3.6178    4.5401     4.169    3.1511  -0.35388   0.15731    5.6679   -2.1184   -2.4507    3.6151

  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_nrldpc_sim
         0         1   0.28713       100     10107       100

>> BPSK_nrldpc_sim
         1      0.92   0.19849        92      6987       100

>> BPSK_nrldpc_sim
         1      0.81   0.19514        81      6869       100

>> BPSK_nrldpc_sim
         2      0.18  0.027841        18       980       100

>> BPSK_nrldpc_sim
         2      0.28  0.037812        28      1331       100

fx >>
```

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)



```
  Columns 11 through 20

      3.6178      4.5401       4.169      3.1511    -0.35388     0.15731      5.6679     -2.1184     -2.4507      3.6151

  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_nrldpc_sim
           0           1     0.28713         100       10107         100

>> BPSK_nrldpc_sim
           1        0.92     0.19849          92        6987         100

>> BPSK_nrldpc_sim
           1        0.81     0.19514          81        6869         100

>> BPSK_nrldpc_sim
           2        0.18    0.027841          18         980         100

>> BPSK_nrldpc_sim
           2        0.28    0.037812          28        1331         100

>> BPSK_nrldpc_sim
           2        0.22    0.037528          22        1321         100

fx >>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

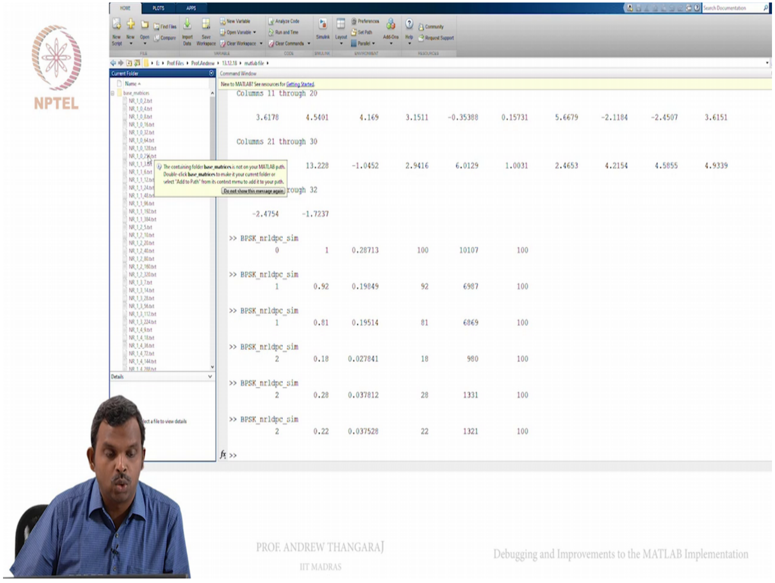
Debugging and Improvements to the MATLAB Implementation

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)



```
Command Window
New to MATLAB? See resources for Getting Started.
  Columns 11 through 20

    3.6178    4.5401     4.169    3.1511   -0.35388   0.15731    5.6679   -2.1184   -2.4507    3.6151

  Columns 21 through 30

              13.228   -1.0452    2.9416    6.0129    1.0031    2.4653    4.2154    4.5855    4.9339

            rough 32

   -2.4754   -1.7237

>> BPSK_nrldpc_sim
         0         1   0.28713       100     10107       100

>> BPSK_nrldpc_sim
         1      0.92   0.19849        92      6987       100

>> BPSK_nrldpc_sim
         1      0.81   0.19514        81      6869       100

>> BPSK_nrldpc_sim
         2      0.18  0.027841        18       980       100

>> BPSK_nrldpc_sim
         2      0.28  0.037812        28      1331       100

>> BPSK_nrldpc_sim
         2      0.22  0.037528        22      1321       100

fx >>
```

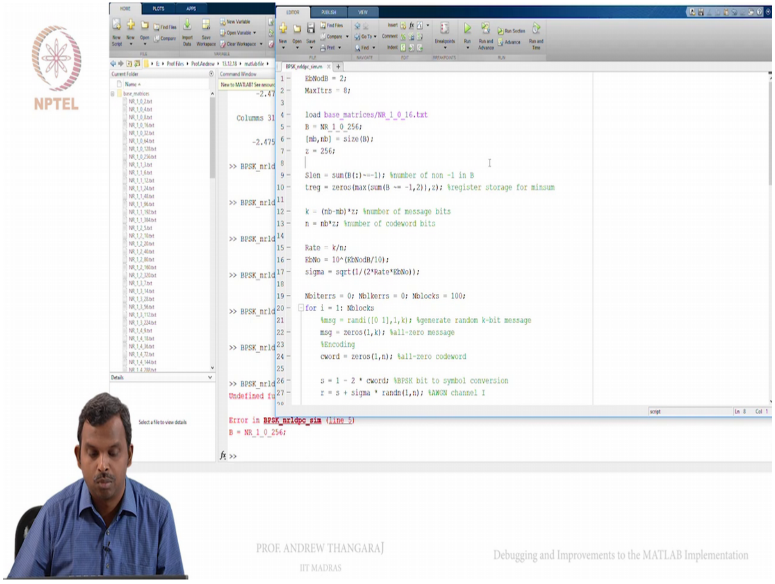
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
EbNodB = 2;
MaxItrs = 8;

load base_matrices/NR_1_0_16.txt
B = NR_1_0_256;
[mb,nb] = size(B);
z = 256;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

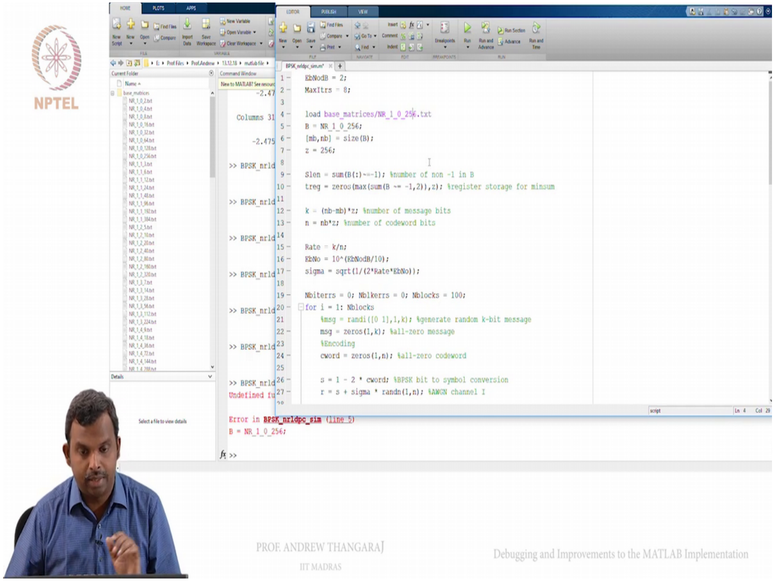
```
>> BPSK_nrld
Undefined fu

Error in BPSK_nrldpc_sim (line 5)
B = NR_1_0_256;

fx >>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

Ok, Ok so I made a mistake here. I should load

(Refer Slide Time: 18:29)



```
EbNodB = 2;
MaxItrs = 8;

load base_matrices/NR_1_0_256.txt
B = NR_1_0_256;
[mb,nb] = size(B);
z = 256;

Slen = sum(B(:)~=-1); %number of non -1 in B
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I
```

```
Error in BPSK_nrldpc_sim (line 5)
B = NR_1_0_256;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

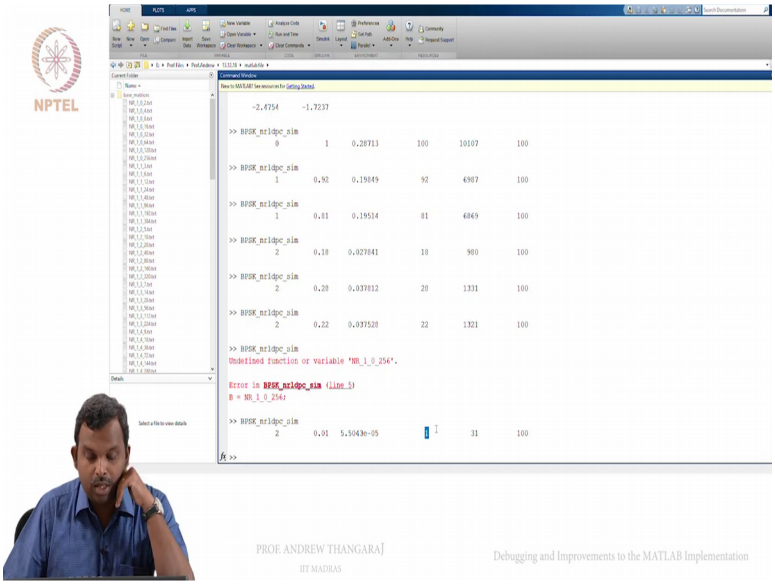
256 here, Ok. So let us see. This is a much larger block length. And let us see if this works correctly. Here you can see it is taking a little bit of time. So generally L D P C codes at larger block lengths will perform much, much better.

Ok so block length of 1000 is a bit low for L D P C 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. 2 d B E b over N 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)



```
     -2.4754     -1.7237

>> BPSK_nrldpc_sim
         0         1    0.28713       100     10107       100

>> BPSK_nrldpc_sim
         1      0.92    0.19849        92      6987       100

>> BPSK_nrldpc_sim
         1      0.81    0.19514        81      6869       100

>> BPSK_nrldpc_sim
         2      0.18   0.027841        18       980       100

>> BPSK_nrldpc_sim
         2      0.28   0.037812        28      1331       100

>> BPSK_nrldpc_sim
         2      0.22   0.037528        22      1321       100

>> BPSK_nrldpc_sim
Undefined function or variable 'NR_1_0_256'.

Error in BPSK_nrldpc_sim (line 5)
B = NR_1_0_256;

>> BPSK_nrldpc_sim
         2      0.01 5.5043e-05         1        31       100

fx >>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Debugging and Improvements to the MATLAB Implementation

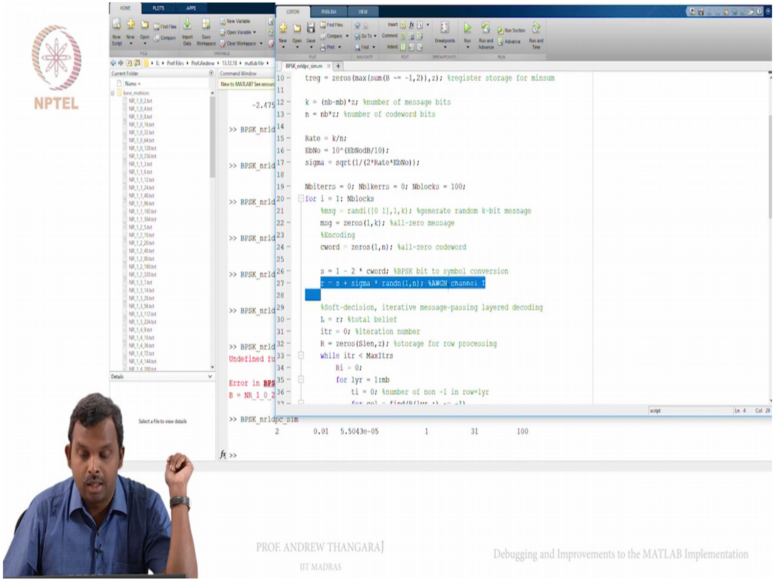
Ok, out of 100, that actually looks quite wrong to me. It should be much, much better than this.

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)

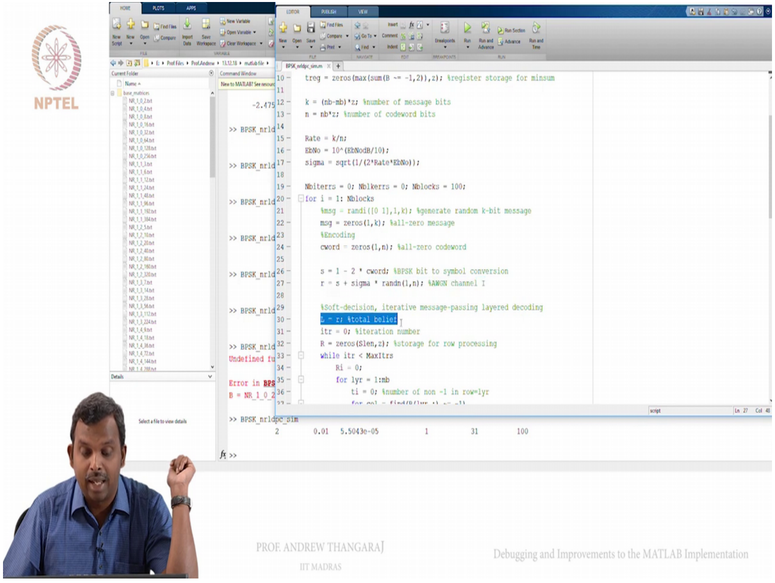


```
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I

    %Soft-decision, iterative message-passing layered decoding
    L = r; %total belief
    itr = 0; %iteration number
    R = zeros(Slen,z); %storage for row processing
    while itr < MaxItrs
        Ri = 0;
        for lyr = 1:mb
            ti = 0; %number of non -1 in row=lyr
```

```
    -2.475

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrld

>> BPSK_nrld
Undefined fu

Error in BPS
B = NR_1_0_2

>> BPSK_nrldpc_sim
    2       0.01    5.5043e-05          1          31          100

fx >>
```

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

(Refer Slide Time: 20:32)



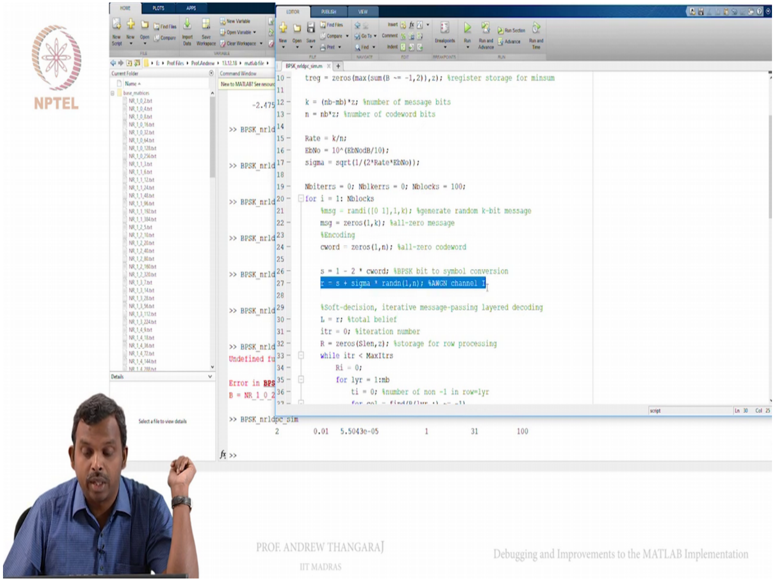
```
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I

    %Soft-decision, iterative message-passing layered decoding
    L = r; %total belief
    itr = 0; %iteration number
    R = zeros(Slen,z); %storage for row processing
    while itr < MaxItrs
        Ri = 0;
        for lyr = 1:mb
            ti = 0; %number of non -1 in row=lyr
```

```
>> BPSK_nrldpc_sim
    2       0.01    5.5043e-05          1          31          100

fx >>
```

L, Ok. So that is not very nice.

So you want to, you want to quantize

(Refer Slide Time: 20:37)



```
treg = zeros(max(sum(B ~= -1,2)),z); %register storage for minsum

k = (nb-mb)*z; %number of message bits
n = nb*z; %number of codeword bits

Rate = k/n;
EbNo = 10^(EbNodB/10);
sigma = sqrt(1/(2*Rate*EbNo));

Nbiterrs = 0; Nblkerrs = 0; Nblocks = 100;
for i = 1: Nblocks
    %msg = randi([0 1],1,k); %generate random k-bit message
    msg = zeros(1,k); %all-zero message
    %Encoding
    cword = zeros(1,n); %all-zero codeword

    s = 1 - 2 * cword; %BPSK bit to symbol conversion
    r = s + sigma * randn(1,n); %AWGN channel I

    %Soft-decision, iterative message-passing layered decoding
    L = r; %total belief
    itr = 0; %iteration number
    R = zeros(Slen,z); %storage for row processing
    while itr < MaxItrs
        Ri = 0;
        for lyr = 1:mb
            ti = 0; %number of non -1 in row=lyr
```

```
>> BPSK_nrldpc_sim
     2       0.01   5.5043e-05          1          31          100
```

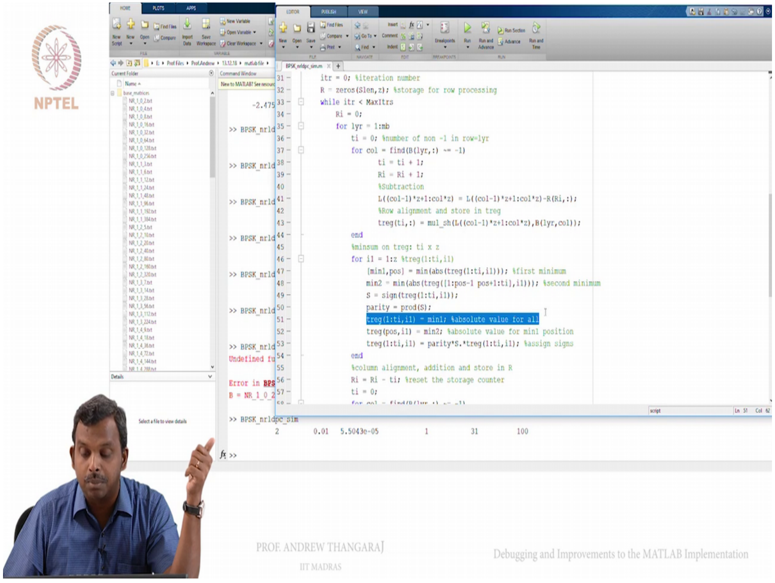
PROF. ANDREW THANGARAJ
IIT MADRAS

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)



```
        itr = 0; %iteration number
        R = zeros(Slen,z); %storage for row processing
        while itr < MaxItrs
            Ri = 0;
            for lyr = 1:mb
                ti = 0; %number of non -1 in row=lyr
                for col = find(B(lyr,:) ~= -1)
                        ti = ti + 1;
                        Ri = Ri + 1;
                        %Subtraction
                        L((col-1)*z+1:col*z) = L((col-1)*z+1:col*z)-R(Ri,:);
                        %Row alignment and store in treg
                        treg(ti,:) = mul_sh(L((col-1)*z+1:col*z),B(lyr,col));
                end
                %minsum on treg: ti x z
                for i1 = 1:z %treg(1:ti,i1)
                    [min1,pos] = min(abs(treg(1:ti,i1))); %first minimum
                    min2 = min(abs(treg([1:pos-1 pos+1:ti],i1))); %second minimum
                    S = sign(treg(1:ti,i1));
                    parity = prod(S);
                    treg(1:ti,i1) = min1; %absolute value for all
                    treg(pos,i1) = min2; %absolute value for min1 position
                    treg(1:ti,i1) = parity*S.*treg(1:ti,i1); %assign signs
                end
                %column alignment, addition and store in R
                Ri = Ri - ti; %reset the storage counter
                ti = 0;
```

```
>> BPSK_nrldpc_sim
     2       0.01   5.5043e-05          1          31          100
```

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.