

**Computer Aided Applied Single Objective Optimization**  
**Dr. Prakash Kotecha**  
**Department of Chemical Engineering**  
**Indian Institute of Science, Guwahati**

**Lecture – 24**  
**Case Study: Production Planning MATLAB Implementation**

Welcome back. In this session, what we will be doing is we will be first coding the Production Planning problem which we have discussed previously and then we will plug it with optimization algorithm right. So, we will be plugging it initially with TLBO. So, the first part of this session is going to focus on developing the fitness function file for the production planning problem right.

(Refer Slide Time: 00:53)

**Case Study: Production Planning Industry**

Sale Price (monetary unit/unit of product)	Product	Process	Capacity (units of product/vr)			Production cost (monetary unit/unit of product)			Investment cost (monetary unit/unit of product)			Raw material used (per unit of product)		
			$l_j$	$m_j$	$h_j$	$C_{ij}$	$C_{mj}$	$C_{hj}$	$V_{ij}$	$V_{mj}$	$V_{hj}$	R1	R2	R3
0.975	T1	P1	50	100	270	50.7	90.1	170.7	55	81.1	131.6	0.948	0	0
		P2	75	150	300	56.8	103.8	196.2	58	85.1	132.4	0.9432	0	0
		P3	77.5	155	310	56.9	103.7	195.7	60.2	86.8	134.1	0.949	0	0
0.975	T2	P4	70	145	290	51.7	97.6	184.8	55.1	83.1	132	0.9546	0	0
		P5	47.5	95	190	38.2	69.8	130.4	43.3	66.8	104.3	0.955	0	0
0.780	T3	P6	40	80	160	38.5	65.2	120.7	66.2	92.8	153.2	1.045	0	0
		P7	40	80	160	31.8	57.1	105.5	40	61.4	95.1	1.05	0	0
0.735	T4	P8	45	90	180	37.8	57.7	94.9	106.6	151.7	231.5	0.5103	0	0
		P9	40	80	160	38.5	65.6	119.1	82.8	125.4	207	0.6289	0	0
0.1450	T5	P10	90	180	360	92.2	159.2	290.9	233.5	390.7	698.7	0.8648	0	0
		P11	90	180	360	86.7	154.1	287.7	185.8	304.5	537.1	0.9546	0	0
		P12	90	180	360	95.8	175	330.9	119	179.4	289.2	0.8265	0	0
		P13	90	180	360	87.5	157.2	294.9	212.3	362.7	657.7	0.7875	0	0
		P14	90	180	360	105.9	196.6	375.2	109.8	164.3	263.1	0.8101	0	0
0.830	T7	P15	90	180	360	93.1	131.1	239.4	221.7	376.1	672.7	0.8782	0	0
		P16	50	100	200	41.4	68.7	117.2	115.5	180.4	287.4	0.815	0	0
		P17	50	100	200	34.9	62	111.6	63.7	100.2	156.3	0.6994	0	0
0.450	T8	P18	60	120	240	36.6	62.1	120.8	23.1	33.2	50.7	0.3784	0	0

T5 P9

So, just to recollect the problem right so, here we will be having few processes right. So, in the slide, we have shown you only 8 products, but there are 24 such products right. So, as we

can see product 1 can be produced by three process. Process 1, process 2, process 2; 2 can be produced by 4 5, 3 can be produced by 6 7 and so on right. So, we will be having till P 54 right. So, 24 products that can be produced by 54 processes; again not all products can be produced by every process right

So, these are the production costs associated with each process. So, process 1 if we operate at low capacity level, we will have a capacity of 70 right; so, 70 units of product per year, so, you can take it as 70 tons per year. Similarly if we operate it at medium level it is 135 tons per year and if we operate it at h j, it is 270 tons per year right. So, the production cost is also given right.

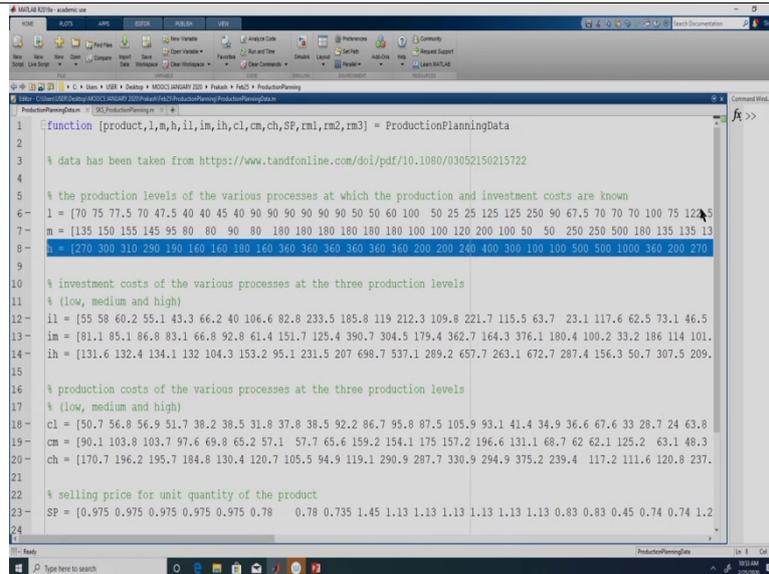
So, if we produce 70, the production cost is 50.7; if you produce 135, the production cost is 90.1. If we produce 270, the production cost is 170.9. So, in between these if we produce. So, if we produce between seventy and 135, then we will have to use the line connecting 70 comma 50.7, 135 comma 90.1 to find out the production cost at any production which is between 70 and 135 right.

So, similarly we will be having this investment cost. So, if we produce 70 units per year the amount of investment cost that will require is 55 monetary units per unit of product. If we produce 135, the investment cost is 81.1. If we produce 270, the investment cost is 131.6 right. So, this is true for all the 54 processes and then we have three types of raw material raw material 1, raw material 2, raw material 3.

So, at least this first 18 process which we have do not consume raw material 1 and raw material 3 right. So, if you remember the previous slides some of the rest of the processes consume R 2 and R 3 right. So, this is the amount of raw material 1 that is required to produce 1 unit of product right. So, if we decide to produce 70 units from process 1, then it is 0.948 into 70.

So, similarly we are also given the selling price right. So, if we sell 1 unit of product T 1, the amount that we get is 0.975. So, if we produce 100, units over here then it is 100 into 0.975. So, our first job is to get this data.

(Refer Slide Time: 03:28)



```
1 function [product,l,m,b,il,im,ih,c1,cm,ch,SP,rm1,rm2,rm3] = ProductionPlanningData
2
3 % data has been taken from https://www.tandfonline.com/doi/pdf/10.1080/030521502150215722
4
5 % the production levels of the various processes at which the production and investment costs are known
6 l = [70 75 77.5 70 47.5 40 40 45 40 90 90 90 90 90 90 50 50 60 100 50 25 25 125 125 250 90 67.5 70 70 100 75 122.5
7 m = [135 150 155 145 95 80 80 90 80 180 180 180 180 180 180 100 100 120 200 100 50 50 250 250 500 180 135 135 13
8 b = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 200 200 240 400 300 100 100 500 500 1000 360 200 270
9
10 % investment costs of the various processes at the three production levels
11 % (low, medium and high)
12 il = [55 58 60.2 55.1 43.3 66.2 40 106.6 82.8 233.5 185.8 119 212.3 109.8 221.7 115.5 63.7 23.1 117.6 62.5 73.1 46.5
13 im = [81.1 85.1 86.8 83.1 66.8 92.8 61.4 151.7 125.4 390.7 304.5 179.4 362.7 164.3 376.1 180.4 100.2 33.2 186 114 101.
14 ih = [131.6 132.4 134.1 132 104.3 153.2 95.1 231.5 207 698.7 537.1 289.2 657.7 263.1 672.7 287.4 156.3 50.7 307.5 209.
15
16 % production costs of the various processes at the three production levels
17 % (low, medium and high)
18 c1 = [50.7 56.8 56.9 51.7 38.2 38.5 31.8 37.8 38.5 92.2 86.7 95.8 87.5 105.9 93.1 41.4 34.9 36.6 67.6 33 28.7 24 63.8
19 cm = [90.1 103.8 103.7 97.6 69.8 65.2 57.1 57.7 65.6 159.2 154.1 175 157.2 196.6 131.1 68.7 62 62.1 125.2 63.1 48.3
20 ch = [170.7 196.2 195.7 184.8 130.4 120.7 105.5 94.9 119.1 290.9 287.7 330.9 294.9 375.2 239.4 117.2 111.6 120.8 237.
21
22 % selling price for unit quantity of the product
23 SP = [0.975 0.975 0.975 0.975 0.975 0.78 0.78 0.735 1.45 1.13 1.13 1.13 1.13 1.13 1.13 0.83 0.83 0.45 0.74 0.74 1.2
24
```

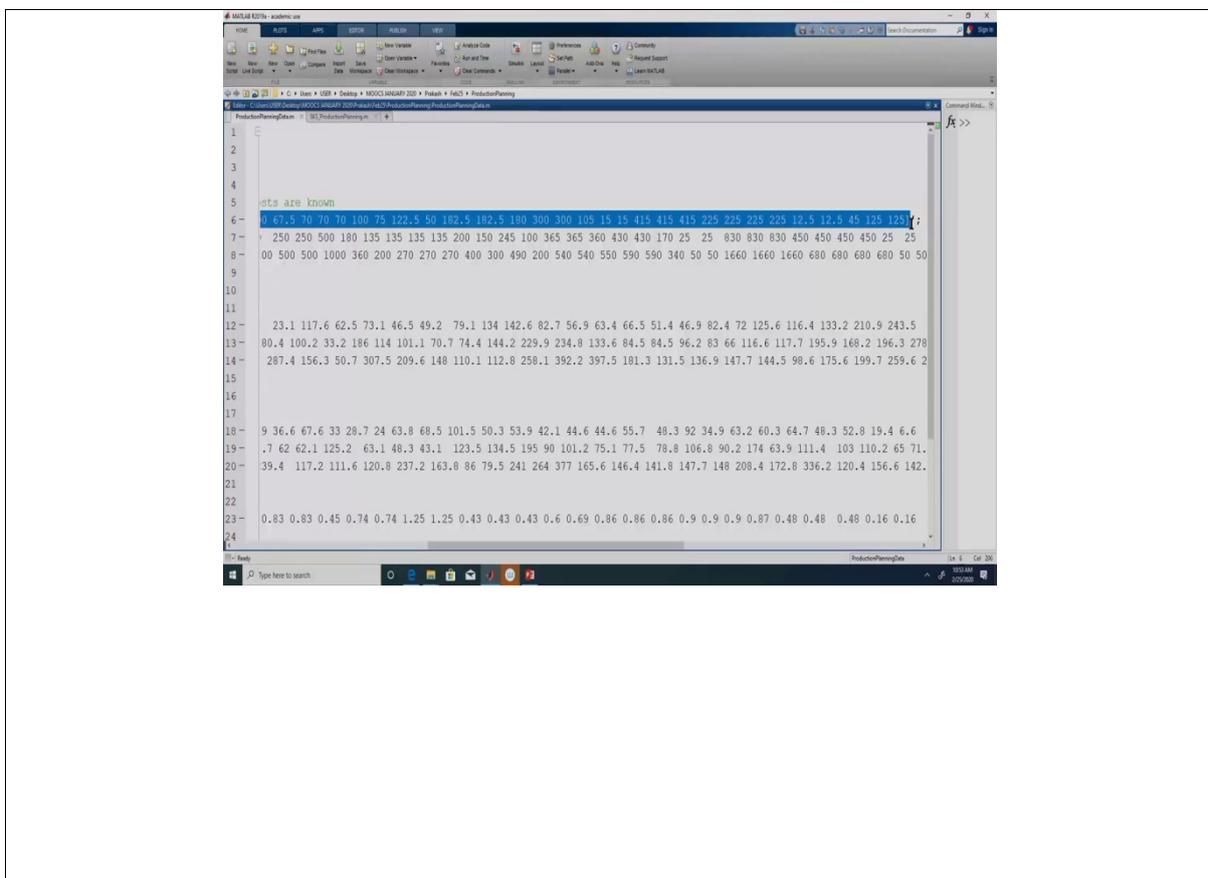
So, what we will do is we will separate the data. So, we will get all the data over here. So, what we have is a function file right. So, the name of the function file is ProductionPlanningData right. So, you can give any other name that you would want right. So, we have given ProductionPlanningData.

So, there is no input to the this function right. So, here if you see on the right hand side, we are not passing any input. This function will only give us the output. So, what we will be doing is we will be calling this function whenever we want to evaluate the fitness function right. So,

to get the data we do not need to supply any input right. So, these are the 3 production levels right.

So, whatever you see in this slide 71, 35, 270 right so, this is 70, the low level the medium level is 135, the high level is 270 right. For the second process, it is 75, 150, 300; so, 74, 150, 300 right.

(Refer Slide Time: 04:21)



Like this we have 54 values in each of this variable right. So, `l` is a column vector right. So, it will have 54 rows and 1 column right. So, we have typed it as row vector. But you have use the transpose over here. Similarly `m` will contain 54 values and `h` will also contain 54 values. So, we have taken this data `l m h` into this file right. Similarly we need to take the investment cost right. So, `il` denotes the investment cost for the low level.

Here if we see the investment cost is over here. So, it is 55 81 and 131.6. So, 55, 81, 136.6 and the second one is 58 81.6 132.4; 58 81.6 132. Just like for each production capacity, we will have a investment cost right. So, if we decide to produce 70, the investment cost is going to be 55. If you decide to produce 135, the investment cost would be 81.1. If we decide to produce 270, the investment cost would be 131.6.

If we decide to produce anything between 70 and 135 the investment cost has to be calculated using this 55 and 81.1. So, that we will be doing later, right now we are only entering the data whatever we have.

(Refer Slide Time: 05:37)

```

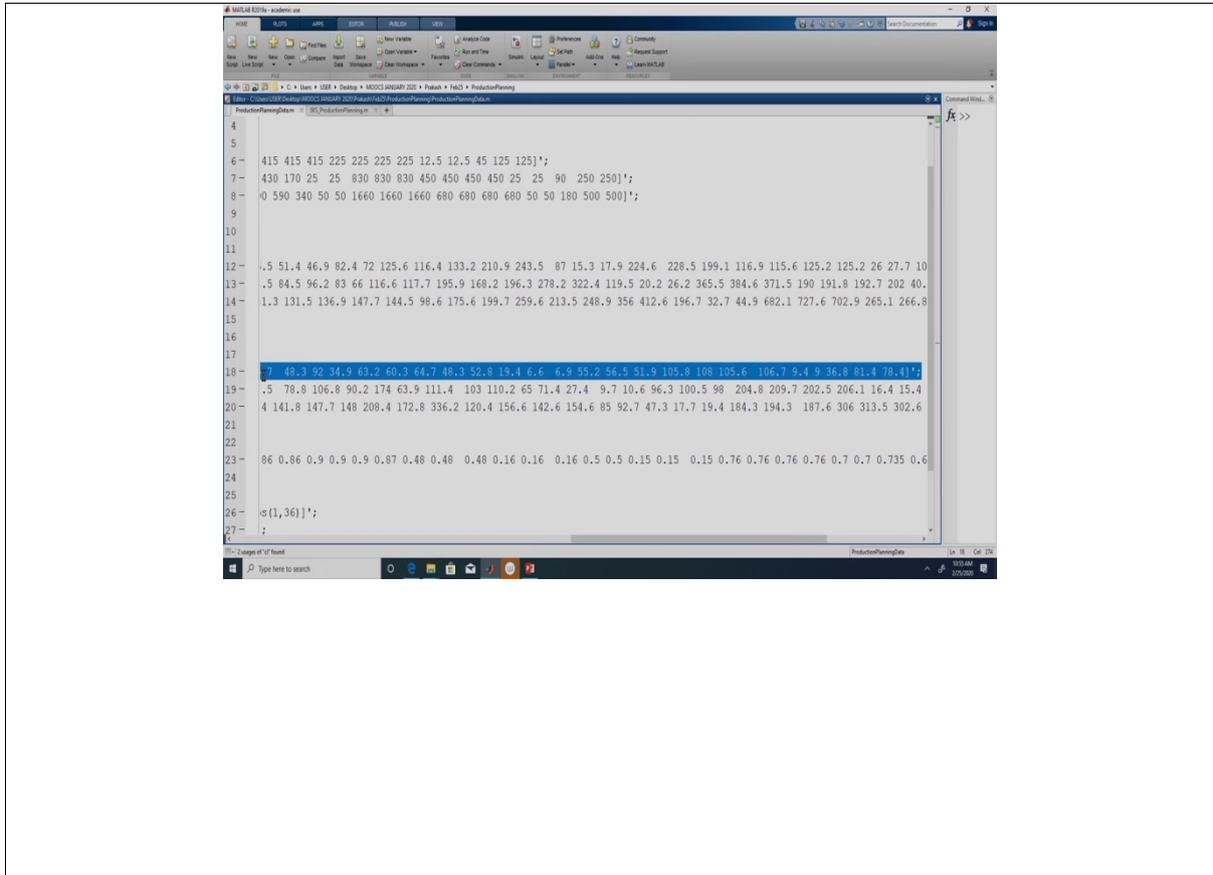
8- h = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 200 240 400 300 100 100 500 500 1000 360 200 270]
9
10 % investment costs of the various processes at the three production levels
11 % (low, medium and high)
12- i1 = [55 58 60.2 55.1 43.3 66.2 40 106.6 82.8 233.5 185.8 119 212.3 109.8 221.7 115.5 63.7 23.1 117.6 62.5 73.1 46.5
13- i2 = [81.1 85.1 86.8 83.1 66.8 92.8 61.4 151.7 125.4 390.7 304.5 179.4 362.7 164.3 376.1 180.4 100.2 33.2 186 114 101.
14- i3 = [131.6 132.4 134.1 132 104.3 153.2 95.1 231.5 207 698.7 537.1 289.2 657.7 263.1 672.7 287.4 156.3 50.7 307.5 209.
15
16 % production costs of the various processes at the three production levels
17 % (low, medium and high)
18- c1 = [50.7 56.8 56.9 51.7 38.2 38.5 31.8 37.8 38.5 92.2 86.7 95.8 87.5 105.9 93.1 41.4 34.9 36.6 67.6 33 28.7 24 63.8
19- c2 = [90.1 103.8 103.7 97.6 69.8 65.2 57.1 57.7 65.6 159.2 154.1 175 157.2 196.6 131.1 68.7 62 62.1 125.2 63.1 48.3
20- c3 = [170.7 196.2 195.7 184.8 130.4 120.7 105.5 94.9 119.1 290.9 287.7 330.9 294.9 375.2 239.4 117.2 111.6 120.8 237.
21
22 % selling price for unit quantity of the product
23- SP = [0.975 0.975 0.975 0.975 0.975 0.78 0.78 0.735 1.45 1.13*ones(1,6) 0.83 0.83 0.45 1.74 0.74 1.25 1.25 0.43
24
25 % raw material required for producing unit quantity of product
26- rm1 = [0.948 0.9432 0.949 0.9546 0.955 1.045 1.05 0.5103 0.6289 0.8648 0.9546 0.8265 0.7875 0.8101
27- rm2 = [zeros(1,24) 0.4678 0.7267 0.393 1.02 1.02 1.02 0.9461 0.9387 0.943 1.06 zeros(1,11) 0.2891
28- rm3 = [zeros(1,37) 6.35 5.928 6.678 0 0 7.867 7.778 7.661 zeros(1,9)];
29
30 % the value of product(i) indicates the product produced by the ith process
31- product = [1 1 1 2 2 2 3 3 4 5 6 6 6 6 6 6 7 7 8 9 9 10 10 11 11 11 12 13 14,

```

So, similarly c1, c2, c3 indicate the production cost corresponding to l, m and h. So, here if we see the production cost is 57.7, 90.1 and 170.7 for process one. So, c1 is 57.7, 90.1, 170.7. Similarly for the second process, it should be 56.8, 103.8 and 196.2 right; so, 56.8 1, 103.8

and 196.2 right. So, this is similar to l, m, h the c l cm ch will be a column vector right. So, at the end we have put this transpose over here. So, it will be a column vector.

(Refer Slide Time: 06:13)



So, what we have done so far. We have entered the production capacity, the investment cost and the production cost right. So, selling price if we see over here so, this we can write it as 0.975, 3 times right. So, because T 1 is produced from process 1, process 2, process 3 and no matter from which process it is produced, the production cost is 0.975. So, that is why we are repeating this 0.975 3 times right; 0.975, 0.975, 0.975 and product 2 is produced from product 4 and product 5 right, but the cost is 0.975.

So, that is why we again have 0.975 twice. So, product 3 is produced from process 6 and 7 and the selling price of product 3 is 0.78 right. So, we will have 0.78 twice because it can be produced by two processes right. So, product T 4 can be produced only from process 8 and

the selling price is 0.735. So, here we have 0.735 just once. So, 1.13 is the selling price of product 6 right and it can be produced by 6 processes right.

So, here we have 1, 2, 3, 4, 5 and 6. So, we have this 6 times right. So, we could do this or we could do 1.13 into 1s of 1 row and 6 columns right. So, this will also give us 1.13, 6 times and then we have a product 7 which is produced from 2 processes right. Cost is 0.83. So, 0.83, 0.83 and we have product 8 which is produced only from process 18 and the selling price is 0.45.

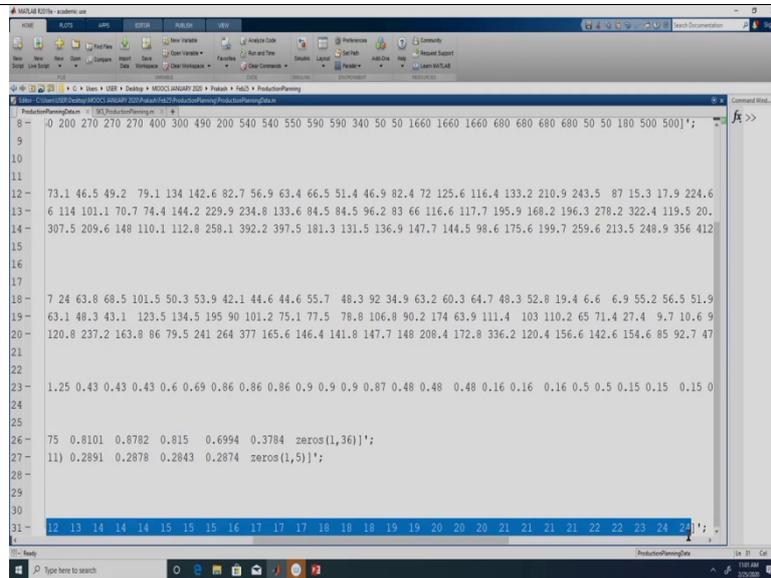
So, we have this 0.44 over here these are the values for the remaining processes. So, you can go back and look at the slides right and then we have these 3 raw materials; raw material 1, raw material 2, raw material 3 right. So, for example, 0.948 0.9432, 0.949 is the amount of raw material required to produce 1 unit of product right. So, that is why we have 0.948, 0.9432 and 0.949 right.

So, similarly we have entered the values for the rest of the processes right. If you look at the entire table, you will see that the first 24 processes do not require raw material two. So, here at least for the 18 you can see. So, for raw material 3, the first 37 processes do not require raw material 3. So, we have put 0 and then the values given are 6.35, 5.92, 6.678 and this last 9 processes also do not require raw material 3. So, we have put that also as 0s.

So, right now what we have entered is the production capacities at which we know the investment cost and the production cost, the selling price of each product that is coming from each process right; the amount of raw material that is required for each of the 54 process. So, that has been entered right. So, the next thing is to code this information that which of the processes are producing which product right.

So, what we have done is we have defined the variable product right. So, for example, this is the first value of product, this is the second value of product, this is the third value of product right.

(Refer Slide Time: 09:29)



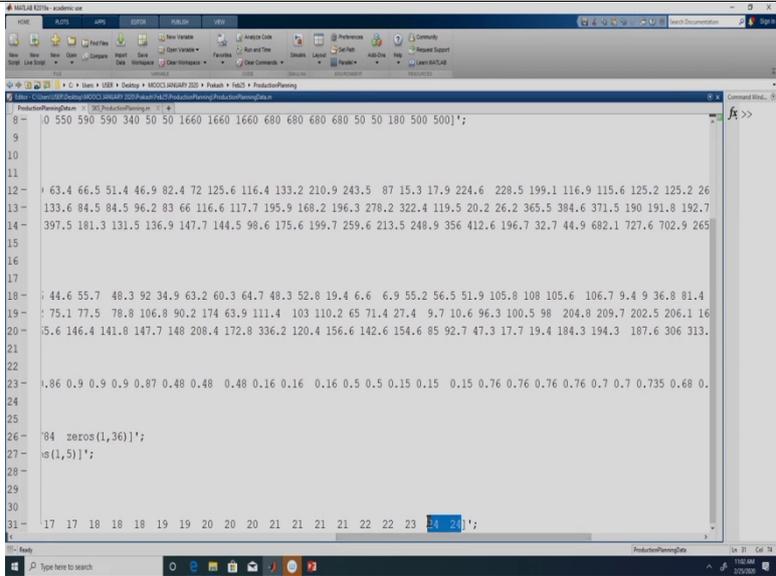
```
ProductionPlanning =  
0 200 270 270 270 400 300 490 200 540 540 550 590 340 50 50 1660 1660 1660 680 680 680 680 50 50 180 500 500];  
9  
10  
11  
12 73.1 46.5 49.2 79.1 134 142.6 82.7 56.9 63.4 66.5 51.4 46.9 82.4 72 125.6 116.4 133.2 210.9 243.5 87 15.3 17.9 224.6  
13 6 114 101.1 70.7 74.4 144.2 229.9 234.8 133.6 84.5 84.5 96.2 83 66 116.6 117.7 195.9 168.2 196.3 278.2 322.4 119.5 20.  
14 307.5 209.6 148 110.1 112.8 258.1 392.2 397.5 181.3 131.5 136.9 147.7 144.5 98.6 175.6 199.7 259.6 213.5 248.9 356 412  
15  
16  
17  
18 7 24 63.8 68.5 101.5 50.3 53.9 42.1 44.6 44.6 55.7 48.3 92 34.9 63.2 60.3 64.7 49.3 52.8 19.4 6.6 6.9 55.2 56.5 51.9  
19 63.1 48.3 43.1 123.5 134.5 195 90 101.2 75.1 77.5 78.8 106.8 90.2 174 63.9 111.4 103 110.2 65 71.4 27.4 9.7 10.6 9  
20 120.8 237.2 163.8 86 79.5 241 264 377 165.6 146.4 141.8 147.7 148 208.4 172.8 336.2 120.4 156.6 142.6 154.6 85 92.7 47  
21  
22  
23 1.25 0.43 0.43 0.43 0.6 0.69 0.86 0.86 0.86 0.9 0.9 0.87 0.48 0.48 0.48 0.16 0.16 0.16 0.5 0.5 0.15 0.15 0  
24  
25  
26 75 0.8101 0.8782 0.815 0.6994 0.3784 zeros(1,36)];  
27 11) 0.2891 0.2878 0.2843 0.2874 zeros(1,5)];  
28  
29  
30  
31 12 13 14 14 14 15 15 16 17 17 18 18 19 19 20 20 21 21 21 22 22 23 24 24];
```

So, there are 54 such values right. So, it indicates what is the product which is produced by that particular process right. So, this is at the first location. So, this indicates about process 1. So, process 1 produces product 1, process 2 produce product 1, process 3 also produces product 1 right; process 4, this is process 4 because this is the fourth value in this vector right. First value, second value, third value and the fourth value so, the fourth and fifth value are 2. So, that means, the fourth process and the fifth process produce product 2 right; this is 6 7 right.

So, the process 6 and process 7 produce product 3. This is 8 right. So, process 8 produces product 4, process 9 produces product 5 right. So, this is 10, 10, 11, 12, 13, 14 and 15. So, till 15th process products 6 is produced that is why we have repeated product 6 6 times right. So, similarly this would be the 16th value and this would be the 17th value right. So, the 16th and

17th value are 7. So, that is why. So, process 16 and process 17 produce product 7. So, that is why 7 is written. So, similarly we have coded for all the processes right.

(Refer Slide Time: 10:44)



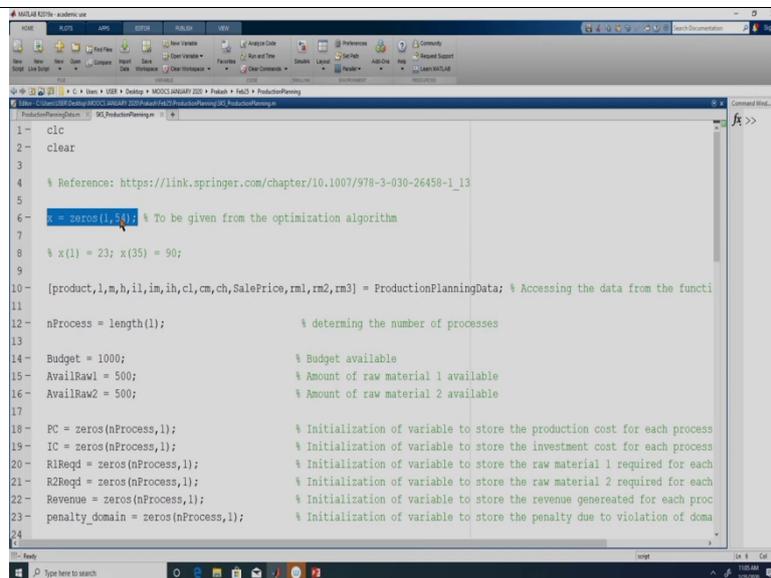
```
0 550 590 590 340 50 50 1660 1660 1660 680 680 680 680 50 50 180 500 500];
9
10
11
12 63.4 66.5 51.4 46.9 82.4 72 125.6 116.4 133.2 210.9 243.5 87 15.3 17.9 224.6 228.5 199.1 116.9 115.6 125.2 125.2 26
13 133.6 84.5 84.5 96.2 83 66 116.6 117.7 195.9 168.2 196.3 278.2 322.4 119.5 20.2 26.2 365.5 384.6 371.5 190 191.8 192.7
14 397.5 181.3 131.5 136.9 147.7 144.5 98.6 175.6 199.7 259.6 213.5 248.9 356 412.6 196.7 32.7 44.9 682.1 727.6 702.9 265
15
16
17
18 44.6 55.7 48.3 92 34.9 63.2 60.3 64.7 48.3 52.8 19.4 6.6 6.9 55.2 56.5 51.9 105.8 108 105.6 106.7 9.4 9 36.8 81.4
19 75.1 77.5 78.8 106.8 90.2 174 63.9 111.4 103 110.2 65 71.4 27.4 9.7 10.6 96.3 100.5 98 204.8 209.7 202.5 206.1 16
20 5.6 146.4 141.8 147.7 148 208.4 172.8 336.2 120.4 156.6 142.6 154.6 85 92.7 47.3 17.7 19.4 184.3 194.3 187.6 306 313.
21
22
23 0.86 0.9 0.9 0.9 0.87 0.48 0.48 0.48 0.16 0.16 0.16 0.5 0.5 0.15 0.15 0.15 0.76 0.76 0.76 0.7 0.7 0.735 0.68 0.
24
25
26 %4 zeros(1,36)';
27 %s(1,5)';
28
29
30
31 17 17 18 18 18 19 20 20 21 21 21 21 22 22 23 24 24];
```

So, we have product 24 which is being produced by process 54 and 53. So, now, that we have this information, we will be passing all this information whenever this function is called right. So, if we call this production planning data, we can get this as output right.

So, product will tell us which process is producing which product l m h are the production capacities, at which the investment cost and the production cost are known; il im ih are the investment cost at l m and h; c l, c m, c h are the production cost at l m and h selling price is the sales price by selling the product from a particular process.

Since we have 54 processes, this we have 54 values; rm 1, rm 2, rm 3 will indicate the raw material 1, 2 and 3 respectively required for each process right. So, this completes the data definition. So, whatever data we have for this problem, we have taken it over here right. So, what we will do is we will not write any piece of code over here. We will just restrict it to the data right. What we will do is we will write another file right which will access this file write and given a x, it will give us fitness function value right.

(Refer Slide Time: 12:02)



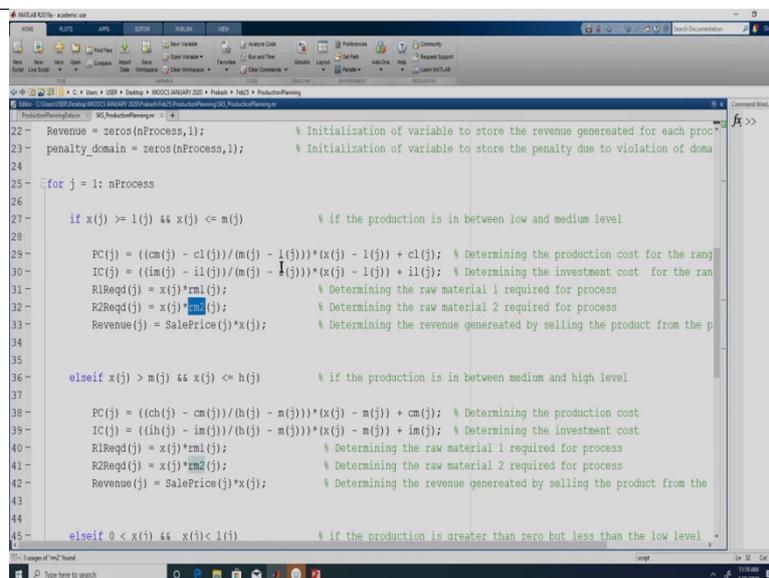
```
1- clc
2- clear
3
4 % Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
5
6 x = zeros(1,54); % To be given from the optimization algorithm
7
8 % x(1) = 23; x(35) = 90;
9
10 [product, l, m, h, il, im, ih, cl, cm, ch, SalePrice, rm1, rm2, rm3] = ProductionPlanningData; % Accessing the data from the functi
11
12 nProcess = length(l); % determining the number of processes
13
14 Budget = 1000; % Budget available
15 AvailRaw1 = 500; % Amount of raw material 1 available
16 AvailRaw2 = 500; % Amount of raw material 2 available
17
18 PC = zeros(nProcess,1); % Initialization of variable to store the production cost for each process
19 IC = zeros(nProcess,1); % Initialization of variable to store the investment cost for each process
20 R1Req1 = zeros(nProcess,1); % Initialization of variable to store the raw material 1 required for each
21 R2Req1 = zeros(nProcess,1); % Initialization of variable to store the raw material 2 required for each
22 Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue generated for each proc
23 penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty due to violation of doma
24
```

So, here initially we are developing a script file. So, remember that first we need to develop the fitness function file right only after we have developed the fitness function file can we solve the optimization problem right. So, we are independently developing the fitness function file without worrying about the optimization. So, clc clear you know right.

So, as we discussed previously, the algorithm will tell us how much quantity of product has to be produced from each process and there are 54 process. When we execute this problem right the algorithm is supposed to give us the decision variables right. So, right now we are not clubbing it into an with an optimization algorithm. So, we will define the x ourselves right.

So, initially we define it as 0s of 1 comma 54; 1 rows and 54 column. So, whatever the decision variables we are going to get from the algorithm is going to be in this form. It will have 1 row and 54 columns right.

(Refer Slide Time: 12:54)



```
Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue generated for each proc
penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty due to violation of doma
24
25 for j = 1:nProcess
26
27     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and medium level
28
29         FC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the production cost for the rang
30         IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the investment cost for the ran
31         R1Reqd(j) = x(j)*rml(j); % Determining the raw material 1 required for process
32         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
33         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the p
34
35     elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between medium and high level
36
37         FC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the production cost
38         IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the investment cost
39         R1Reqd(j) = x(j)*rml(j); % Determining the raw material 1 required for process
40         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
41         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the
42
43     elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but less than the low level
44
45
```

So, then what we are doing is we are accessing that function which we have written previously right; so, production planning data. So, this we are accessing over here right. So, we will get all necessary information over here and we do not need to copy paste this data over here right. So, we can just access this file. So, that way we can keep the calculation part separate from

the data part right. So, once we have this, what we will first do is calculate the number of processes involved right. So, that can be determined from the length of any of this vectors right.

So, all these are vectors which will have uniform number of rows at least in this case where in we know the data for 54 processes right. So, we can measure the length of any of this variable and determine the number of process right. So, let us assume that the amount of budget that is available to us is 1000, the amount of raw material 1 which is available to us as 500 and the amount of raw material 2 that is available to us is 500.

So, whatever production plan we come up with right it should not exceed these values right. For any production plan if the investment cost is going to be greater than 1000, then it is not going to be a feasible plan right. So, we will incur penalty. Similarly for raw material 1 and raw material 2, it has to be less than 500 right.

So, here line 18 to 2, 3 we are defining 6 variables right; 6 variables. This variable PC will contain the production cost for each process. Remember this PC is different from this cl, cl also contains production cost, but this is the production cost of the production indicated by l right. But when we are solving an optimization problem, it is not necessary that we will be either producing l m or h right.

So, we maybe even be producing between l and m or between m and h or we may even choose not to produce it right. So, for whatever amount we are deciding to produce we need to calculate the production cost. So, that is why we are defining this variable pc right. So, for each process, we will have a production cost. Initially we are defining it with 0s; 0s of number of rowses n process which is 54 in this case and 1 column right.

So, similarly for each process, we are finding out what would be the investment cost. So, il, im, ih indicates the investment cost at l, m and h right, but we might have a production which is not necessarily l m h it may be 0 or between l and h right. So, for that we need to calculate

the investment cost accordingly. So, that will be the investment cost over here; raw material 1 required and raw material 2 required.

So, these two variables will store the amount of raw material one that is required for each process, amount of raw material 2 that is required for each process. So, we will be using this variable revenue to store the amount of revenue, we get from each process right and then will sum it up to find out the total revenue, but what is the revenue obtained from each process will be calculated and stored in this vector revenue right. So, again it has 54 values for this data.

So, then if you remember, we also have a domain whole constraint right. So, any value has to be greater than or equal to 1 or it has to be less than or equal to h or it can be 0 right. So, if it is greater than 0, but less than 1 right then we will have to assign a penalty. So, as discussed earlier what we will be doing is if something is greater than 0, but less than 1; we will assign a penalty of  $10^5$  right.

So, but we are not going to assign it for every variable we are only going to assign it for variable which violates. So, what we are doing is, we are defining a variable penalty underscore domain right. Initially it is having a values of 0. So, whichever variable is going to violate that constraint the domain whole constraint, we will be assigning a value of  $10^5$  for that particular process. So, these are just initializations; this fixed variables are just initializations right.

So, now that we have defined the necessary variables. We can calculate the required values right. So, what we are doing is we are running this loop right for j is equal to 1 to n process because this vector x is going to contain 54 values right.

So, for each of the value, we need to see whether it is within the permissible domain or not if it is not within the permissible domain, we are going to assign a penalty. If it is going to be in the permissible domain, we are going to calculate the cost accordingly. This for loop ensures that we calculate everything for each of the process.

(Refer Slide Time: 17:25)

### Determination of cost between different levels

<p><b>Production cost between lower and medium level</b></p> <p><math>x = X</math>  <math>x_1 = l, x_2 = m</math>  <math>y_1 = C_l, y_2 = C_m</math></p> $C = C_l + \frac{C_m - C_l}{l - m}(X - l)$	<p><b>Investment cost between lower and medium level</b></p> <p><math>x = X</math>  <math>x_1 = l, x_2 = m</math>  <math>y_1 = V_l, y_2 = V_m</math></p> $V = V_l + \frac{V_m - V_l}{l - m}(X - l)$
<p><b>Production cost between medium and higher level</b></p> <p><math>x = X</math>  <math>x_1 = m, x_2 = h</math>  <math>y_1 = C_m, y_2 = C_h</math></p> $C = C_m + \frac{C_h - C_m}{m - h}(X - m)$	<p><b>Investment cost between medium and higher level</b></p> <p><math>x = X</math>  <math>x_1 = m, x_2 = h</math>  <math>y_1 = V_m, y_2 = V_h</math></p> $V = V_m + \frac{V_h - V_m}{m - h}(X - m)$
<p><b>Production cost between lower and medium level</b></p> <p><math>x(j) = X(j)</math>  <math>x_1 = l(j), x_2 = m(j)</math>  <math>y_1 = C_l(j), y_2 = C_m(j)</math></p> $C(j) = C_l(j) + \frac{C_m(j) - C_l(j)}{l(j) - m(j)}(X(j) - l(j))$	<p><b>Investment cost between lower and medium level</b></p> <p><math>x(j) = X(j)</math>  <math>x_1 = l(j), x_2 = m(j)</math>  <math>y_1 = V_l(j), y_2 = V_m(j)</math></p> $V(j) = V_l(j) + \frac{V_m(j) - V_l(j)}{l(j) - m(j)}(X(j) - l(j))$
<p><b>Production cost between medium and higher level</b></p> <p><math>x(j) = X(j)</math>  <math>x_1 = m(j), x_2 = h(j)</math>  <math>y_1 = C_m(j), y_2 = C_h(j)</math></p> $C(j) = C_m(j) + \frac{C_h(j) - C_m(j)}{m(j) - h(j)}(X(j) - m(j))$	<p><b>Investment cost between medium and higher level</b></p> <p><math>x(j) = X(j)</math>  <math>x_1 = m(j), x_2 = h(j)</math>  <math>y_1 = V_m(j), y_2 = V_h(j)</math></p> $V(j) = V_m(j) + \frac{V_h(j) - V_m(j)}{m(j) - h(j)}(X(j) - m(j))$

So, these equations we have derived previously are correct. So, this is nothing, but if the production cost is between lower and medium level, we will have to use this equation. If the production cost is between medium and high level, we need to use this equation. If the investment cost is between lower and medium level, we need to use this equation. This is for investment cost; these two equations are for investment cost.

This is valid between low and medium level, this is valid between medium and high level. Similarly, production cost is valid between lower and medium level, and this one is valid between medium and high level. This is for one right. So, this is for one particular process. So, for the  $j$ th process, it will be the same thing instead of this  $cl$  we are going to have  $cl_j$ . So, for every process, these values are different;  $cl, cm, ch$  values are going to be different;  $l, m, h$  are going to be different for each process.

So, need to quote these four equations for as we have discussed the production cost between  $l$  and  $m$  is given by this expression right. So, how to get this expression that we have seen a little while earlier right. This is the investment cost for a process which produces between  $l$  and  $m$  right greater than or equal to  $l$  and less than or equal to  $m$ . This 2 lines will help us to calculate the production and investment cost right that is because the production and investment cost in this region right varies linearly. So, that is why we are using these two equations right.

So, the raw material 1 that is required is the amount that is produced right multiplied by the amount that is required to produce 1 unit right. So, this is given per unit right. So, if we produce 1 unit of product 1, the amount that we require is 0.94 unit for process 1. If we decide to produce 1 unit of product right, the amount that is required is 0.9432 if we decide to produce.

Let us say 80 units over here for product 2, then the amount total amount of raw material 1 that is required  $0.94 \times 2$  into 80 right. So, that is what we are doing over here. So, the amount of production multiplied by how much of raw material 1 is required for producing 1 unit right.

So, similarly we calculate for rm 2 right. So, these two equations are similar except for here it is rm 1, the amount of raw material 1 that is required and here the amount of raw material 2 that is required. So, if this value is 0 so, as may number for many processes raw material 2 is not required. So, this value will be 0, this line 32 would give us 0 right. Similarly here we are calculating the revenue. So, we know the sales price. So, for each product, we have been given the selling price. What we are doing is the amount that is produced which is  $x_j$ , we are multiplying it with selling price for 1 unit.

So, this will give us the revenue for process  $j$  right. This equations are valid for production greater than or equal to  $l$  and less than or equal to  $m$  right. If it is greater than  $m$  right and less than equal to  $h$ , we will calculate accordingly right. So, these 3 equation are the same, it does not matter what the range of  $x$  is as long as it is between  $l$  and  $h$  right.

So, these two equations change. Here we had the  $cm$  minus  $cl$  in the numerator. Here we have  $ch$  minus  $cm$  in the denominator here we had  $m$  minus  $l$  here we have  $h$  minus  $m$  and  $j$ . So, again this is the same set of equation that we have derived previously. So, right now what we have done is given any production which is greater than or equal to  $l$  and less than or equal to  $h$ , we can calculate the production cost investment cost raw material 1, raw material 2 that is required the revenue right.

(Refer Slide Time: 21:10)

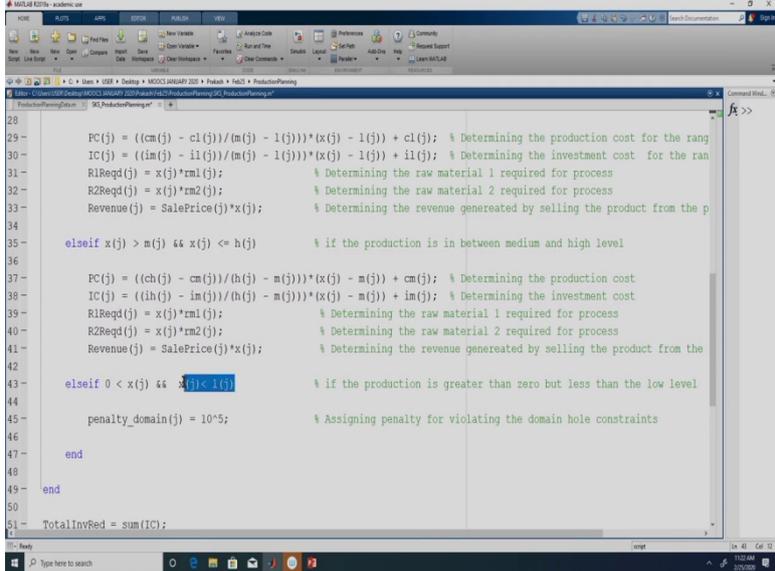
```

22- Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue generated for each process
23- penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty due to violation of domain
24
25- for j = 1:nProcess
26
27-     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and medium level
28
29-         FC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the production cost for the range l to m
30-         IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the investment cost for the range l to m
31-         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required for process
32-         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
33-         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the process
34-
35-     elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between medium and high level
36
37-         FC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the production cost for the range m to h
38-         IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the investment cost for the range m to h
39-         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required for process
40-         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
41-         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the process
42-         penalty_domain(j) = 0; % Determining the penalty due to violation of domain
43-
44-     elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but less than the low level
45-

```

And if the production falls between  $l$  and  $h$ , then the domain whole is violated right. So, since this penalty domain is already 0, we are not writing here penalty domain of  $j$  equal to 0. If it is in the range right between  $l$  and  $m$  or if it is in the range  $m$  and  $h$  the penalty is 0, but we do not need to write this line because the penalty is anyway initially assigned as 0. So, only in those cases where it is violating will we go and assign a penalty.

(Refer Slide Time: 21:40)



```
28
29 PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the production cost for the rang
30 IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the investment cost for the rang
31 R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required for process
32 R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
33 Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the p
34
35 elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between medium and high level
36
37 PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the production cost
38 IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the investment cost
39 R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required for process
40 R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
41 Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the product from the
42
43 elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but less than the low level
44
45 penalty_domain(j) = 10^5; % Assigning penalty for violating the domain hole constraints
46
47 end
48
49 end
50
51 TotalInvRed = sum(IC);
```

So, the third condition is this one right. So, this is between l and m, this is between m and h and this is x is greater than 0 right, but it is less than l right. So, in that what we need to do is we cannot calculate the production cost. So, for example, consider this process 9. So, the production cost from 40 to 80, we know how to calculate and if it is 0 right, then there is no problem the production cost is 0, the investment cost is 0, the raw material 1 that is required is 0, raw material 2 that is required is 0.

And no matter what the selling price, the revenue earned is 0 because the production itself is 0 quantity right. But if the production is let us say, 10 as far as the algorithm concerned, the lower bound is 0, the upper bound is 160. So, it can give any value between 0 and 160; let us assume it gives a value 10.

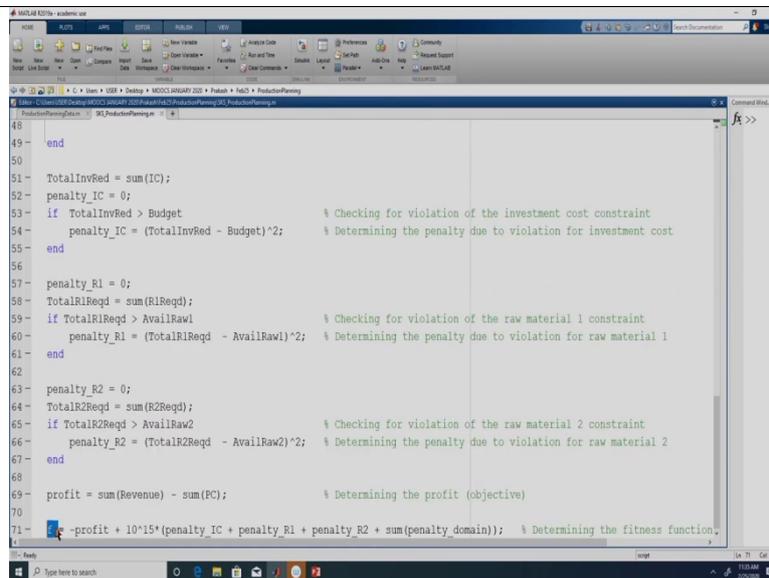
So, if it gives a value of 10, it violates the domain whole constraints, it is greater than 0, but less than 40 right. So, we cannot calculate the production cost investment cost, the revenue or anything. Since this production cost and investment cost is anyway assigned as 0, we do not do anything to the production cost and investment cost.

Similarly, we do not calculate raw material 1, raw material 2 because it is 0. So, similarly revenue is also 0. If it falls in this range right so, what we will do is so, but in this case we need to assign the penalty. What we are doing over here is the variable penalty domain, the  $j$ 'th value in that right. So, this is going to be a vector penalty underscore domain, it is going to have 54 values for this example right so, the  $j$ 'th process. So, whichever process is violating, we are assigning a value of 10 power 5 to that right.

So, this section so, line 25 to 47 will help us to calculate the production cost, the investment cost, the raw material 1, the raw material 2, the revenue and it also assigns penalty wherever the domain whole constraints is violated right. So, now, that we have calculated for all the 54 process, we need to check for the additional 3 constraints. Remember we had four types of constraint; one is domain whole constraint which is taken care over here right. So, assign penalty whenever it is violated, but there are 3 additional contains right.

So, we need to see whether those constraints are satisfied or not if they are satisfied, we do not assign penalty. But if they are not satisfied, we will have to assign a penalty right.

(Refer Slide Time: 24:07)



```
49- end
50
51- TotalInvRed = sum(IC);
52- penalty_IC = 0;
53- if TotalInvRed > Budget % Checking for violation of the investment cost constraint
54-     penalty_IC = (TotalInvRed - Budget)^2; % Determining the penalty due to violation for investment cost
55- end
56
57- penalty_R1 = 0;
58- TotalR1Reqd = sum(R1Reqd);
59- if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw material 1 constraint
60-     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to violation for raw material 1
61- end
62
63- penalty_R2 = 0;
64- TotalR2Reqd = sum(R2Reqd);
65- if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw material 2 constraint
66-     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to violation for raw material 2
67- end
68
69- profit = sum(Revenue) - sum(PC); % Determining the profit (objective)
70
71- -profit + 10*15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain)); % Determining the fitness function.
```

So, in line 51 what we are doing is we are calculating the total investment cost that is required. So, this variable total invRed determines the total production cost. So, we know the production cost of each process right that is in the vector IC. So, here what we are doing is we are merely summing that vector so, that will tell us what is the total investment cost that is required right.

In line 52, we are initializing the variable penalty underscore IC. So, this variable will contain the penalty assigned for violation in the budget constraints right. So, initially we are assigning a value of 0 to it right. So, then we are making a check over here. So, what is the total investment that is required is given by this and what is the budget that is available is given over here right. So, we have defined budget as 1000 in this case. So, if this condition is satisfied, we

are actually violating a constraints that the total investment which is required is greater than budget right.

So, we are violating that constraints. So, if you are violating the constraints, we need to assign penalty right. So, here we are calculating penalty underscore IC incurred because of the violation in the budget constraint right. So, penalty underscore IC is whatever investment is required minus the budget right. So, let us say the investment required is 1200 right.

So, this will be 1200 minus budget. So, budget in this case is 1000. So, it will be 1200 minus 1000 the whole square. So, that is the penalty which will be assigned for violating the investment cost constraint. So, on the contrary if the value of total inverse Red is 800 let us say and the budget is 1000 right. So, what we are saying is amount of budget that is available is 1000 how much we are utilizing is only 800 right. So, it does not satisfy this condition.

So, if it does not satisfy this condition means the solution is not violating. If it is not violating, then no penalty is to be assigned. In that case penalty underscore IC will remain as 0. Similarly right in this section, we calculate the penalty which is incurred if the solution violates the constraints related to raw material 1 right. So, the amount of raw material 1 which is available is AvailRaw1 what is required is summation of the individual required.

So, remember we have calculated for each process over here for each of the j process as to how much of raw material 1 is required right. So, if you sum all of those values that will tell us how much of total raw material 1 is required and then we are checking for the constraint over here right. So, if it is less or equal, then we do not need to assign a penalty; if it is greater we need to assign a penalty over here, this line 57 to 61 is for assigning the penalty for violating the constraint of the raw material 1.

So, if this is similar to line 57 to 61. Here we are checking for the violation in raw material 2. So, now, what we have done is we have calculated the production cost for each process, we have calculated the revenue earned from each process and we have also checked for the 4 types of constraint right. So, the constraint on raw material 1, the constraint on raw material 2,

the constraint on investment cost as well as the domain whole constraint right. So, now, the objective function was revenue minus production cost right.

So, revenue is a vector which contains the revenue earned from each of the process right. So, if we sum this, it will tell us the total revenue which is earned right. So, PC contains the production cost for each process. If we sum it up, it tells us the total production cost. So, this is the total revenue which we earned minus the total production cost right. So, that is our profit right. Since our problem is a maximization problem right and all the algorithms which we wrote are minimization algorithm. So, what we are doing is we are converting our maximization problem into minimization problem by writing minus profit over here right.

So, as we discussed earlier a maximization problem can be converted to minimization problem by multiplying the objective function with the negative sign right. So, this minus profit is because of that right and then we need to add the total penalty; penalty underscore IC is the violation due to the investment cost, penalty underscore R 1 is the penalty incurred for violating the raw material 1 constraints, penalty underscore R 2 is the amount of penalty incurred for violating the constraint on raw material 2 right and penalty underscore domain is a vector right. So, for all those processes where in we have violated, we need to sum the penalty.

So, the total penalty is sum of penalty underscore domain. So, this will be a scalar right penalty underscore R will be scalar, penalty underscore IC will be a scalar right where as this penalty underscore domain is a vector right. So, that is why we are summing it up right.

So, this part right is the total penalty right and we are multiplying it by a very large value right the penalty factor so, as to get the fitness function value right. So, this completes the determination of fitness function right. So, if you are given a solution so, this file can help us to find out the profit and the violation in all the constraint if any right.

So, if there are no violation; if a solution satisfies all the constraint all of this would be 0 right and the fitness e function would be nothing, but the objective function right. But if some of the variables violate the constraints right, then an appropriate penalty is calculated and added to

the objective function. Now that we have coded the objective function file or the fitness function file in this case because we have constraints right. Let us actually run it in debug mode right so that we can make sure that it is working fine before plugging it with any optimization algorithm right.

(Refer Slide Time: 30:02)

The screenshot shows the MATLAB R2019a IDE with a script editor on the left and a Command Window on the right. The script editor contains the following code:

```

1 - clc
2 - clear
3
4 % Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
5
6 x = zeros(1,54); % To be given from the optimization algorithm
7
8 % x(1) = 23; x(35) = 90;
9
10 [product, l, m, h, il, im, ih, cl, cm, ch, SalePrice, rml, rm2, rm3] = ProductionPlanningData; % Acco
11
12 nProcess = length(l); % determining the number of processes
13
14 Budget = 1000; % Budget available
15 AvailRaw1 = 500; % Amount of raw material 1 available
16 AvailRaw2 = 500; % Amount of raw material 2 available
17
18 PC = zeros(nProcess,1); % Initialization of variable to store the pr
19 IC = zeros(nProcess,1); % Initialization of variable to store the in
20 R1Reqd = zeros(nProcess,1); % Initialization of variable to store the ra
21 R2Reqd = zeros(nProcess,1); % Initialization of variable to store the ra
22 Revenue = zeros(nProcess,1); % Initialization of variable to store the re
23 penalty_domain = zeros(nProcess,1); % Initialization of variable to store the pe
24

```

The Command Window displays the following output:

```

K>> whos
Name      Size      Bytes
-----
x         1x54      432

K>> whos
Name      Size      Bytes
-----
SalePrice 54x1      54
ch         54x1      54
cl         54x1      54
cm         54x1      54
h          54x1      54
ih         54x1      54
il         54x1      54
im         54x1      54
l          54x1      54
m          54x1      54
product   54x1      54
rml        54x1      54
rm2        54x1      54
rm3        54x1      54
x          1x54      432

```

So, let me put up breakpoint over here. Let me execute this all right. So, right now remember we are not running the optimization problem, we are merely checking whether this file is working fine or not. So, this x is supposed to come from the algorithm right, but since we are not plugging it with any optimization algorithm as of now, we are initializing it with 0s right. So, for 0s if you think about it, then all the decision variables are 0. So, what we are saying is there is no production from any of the process. So, this solution will not violate the domain constraint right because is permissible.

Similarly the consumption of raw materials will be 0 for this solution. Similarly the investment cost for the solution will also be 0 and the production cost will also be 0 and the revenue will also turn out to be 0. So, basically what the fitness function what we are expecting is 0 right. So, let us see if that is we are getting that properly or not; if you get that properly then at least for that solution, we are not having problem, then we will check for couple of other x values and then we will plug it with an optimization algorithm.

Here we are just calling this function production planning data. So, it will get us all the datas. So, here if we see whos, there is just x right. So, now if we look at this whos, then we have got all the data right.

(Refer Slide Time: 31:24)

The screenshot shows a MATLAB script in the Editor and its corresponding Command Window output. The script defines parameters for a production planning problem, including budget, raw material availability, and process requirements. The Command Window displays the resulting values for these parameters.

```

1 - clc
2 - clear
3
4 % Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
5
6 x = zeros(1,54); % To be given from the optimization algorithm
7
8 % x(1) = 23; x(35) = 50;
9
10 [product, l, m, h, il, im, ih, c1, cm, ch, SalePrice, rml, rm2, rm3] = ProductionPlanningData; % Acce
11
12 nProcess = length(l); % determining the number of processes
13
14 Budget = 1000; % Budget available
15 AvailRaw1 = 500; % Amount of raw material 1 available
16 AvailRaw2 = 500; % Amount of raw material 2 available
17
18 PC = zeros(nProcess,1); % Initialization of variable to store the pr
19 IC = zeros(nProcess,1); % Initialization of variable to store the in
20 R1Reqd = zeros(nProcess,1); % Initialization of variable to store the ra
21 R2Reqd = zeros(nProcess,1); % Initialization of variable to store the ra
22 Revenue = zeros(nProcess,1); % Initialization of variable to store the re
23 penalty_domain = zeros(nProcess,1); % Initialization of variable to store the pe
24

```

Command Window Output:

```

172.8000
336.2000
120.4000
156.6000
142.6000
154.6000
85.0000
92.7000
47.3000
17.7000
19.4000
184.3000
194.3000
187.6000
306.0000
313.5000
302.6000
308.1000
28.4000
27.3000
118.7000
275.5000
277.0000

```

So, cl; so, this is cl, cm, ch right. So, if you want, we can just do c l, c m, c h l m h right i l, i m i h.

(Refer Slide Time: 31:42)

Microsoft Excel - Workbook

Columns 1 through 8

1									
2									
3									
4									
5		70.00	135.00	270.00	51.70	103.10	170.70	55.00	81.10
6		75.00	150.00	300.00	56.80	103.80	196.20	58.00	85.10
7		77.50	155.00	310.00	56.90	103.70	195.70	60.20	86.80
8		70.00	145.00	290.00	51.70	97.60	184.80	55.10	83.10
9		47.50	95.00	190.00	38.20	69.80	130.40	43.30	66.80
10		40.00	80.00	160.00	38.50	65.20	120.70	66.20	92.80
11		40.00	80.00	160.00	31.80	57.10	105.50	40.00	61.40
12		45.00	90.00	180.00	37.80	57.70	94.90	106.60	151.70
13		40.00	80.00	160.00	38.50	65.60	119.10	82.80	125.40
14		90.00	180.00	360.00	92.20	159.20	290.90	233.50	390.70
15		90.00	180.00	360.00	86.70	154.10	287.70	185.80	304.50
16		90.00	180.00	360.00	95.80	175.00	330.90	119.00	179.40
17		90.00	180.00	360.00	87.50	157.20	294.90	212.30	362.70
18		90.00	180.00	360.00	105.90	196.60	375.20	109.80	164.30
19		90.00	180.00	360.00	93.10	131.10	239.40	221.70	376.10
20		50.00	100.00	200.00	41.40	68.70	117.20	115.50	180.40
21		50.00	100.00	200.00	34.90	62.00	111.60	63.70	100.20
22		60.00	120.00	240.00	36.60	62.10	120.80	23.10	33.20
23		100.00	200.00	400.00	67.60	125.20	237.20	117.60	186.00
24		50.00	100.00	300.00	33.00	63.10	163.80	62.50	114.00

```
matlab R2019a academic use
K&T> format bank
K&T> [l m h cl cm ch il im ih]

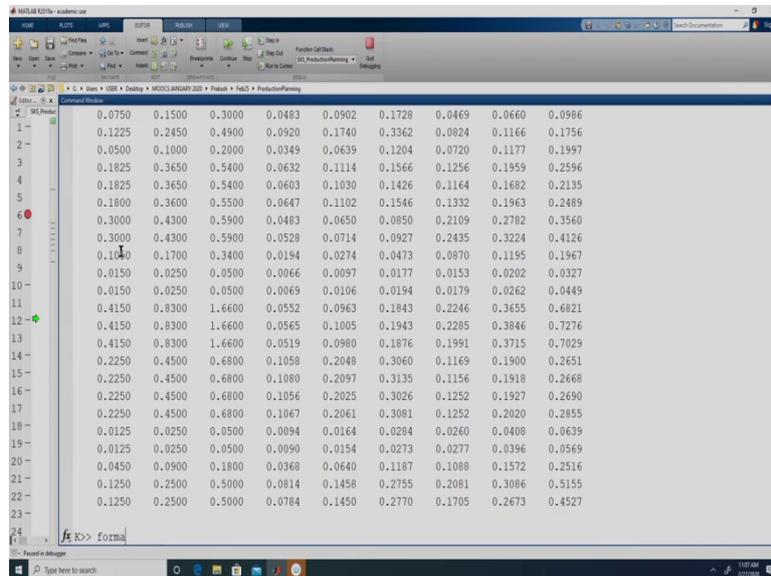
ans =

Columns 1 through 8

    100.00    135.00    270.00    50.70    90.10    170.70    55.00    81.10
    75.00    150.00    300.00    56.80    103.80    196.20    58.00    85.10
    77.50    155.00    310.00    56.90    103.70    195.70    60.20    86.80
    70.00    145.00    290.00    51.70    97.60    184.80    55.10    83.10
    47.50    95.00    190.00    38.20    69.80    130.40    43.30    66.80
    40.00    80.00    160.00    38.50    65.20    120.70    66.20    92.80
    40.00    80.00    160.00    31.80    57.10    105.50    40.00    61.40
    45.00    90.00    180.00    37.80    57.70    94.90    106.60    151.70
    40.00    80.00    160.00    38.50    65.60    119.10    82.80    125.40
    90.00    180.00    360.00    92.20    159.20    290.90    233.50    390.70
    90.00    180.00    360.00    86.70    154.10    287.70    185.80    304.50
    90.00    180.00    360.00    95.80    175.00    330.90    119.00    179.40
    90.00    180.00    360.00    87.50    157.20    294.90    212.30    362.70
    90.00    180.00    360.00    105.90    196.60    375.20    109.80    164.30
    90.00    100.00    200.00    93.10    131.10    239.40    221.70    376.10
    50.00    100.00    200.00    41.40    68.70    117.20    115.50    180.40
    50.00    100.00    200.00    34.90    62.00    111.60    63.70    100.20
```

If you look at this data so, this is the table which we have been seeing right.

(Refer Slide Time: 31:48)



The screenshot shows a Microsoft Excel spreadsheet with the following data:

1	0.0750	0.1500	0.3000	0.0483	0.0902	0.1728	0.0469	0.0660	0.0986
2	0.1225	0.2450	0.4900	0.0920	0.1740	0.3362	0.0824	0.1166	0.1756
3	0.0500	0.1000	0.2000	0.0349	0.0639	0.1204	0.0720	0.1177	0.1997
4	0.1825	0.3650	0.5400	0.0632	0.1114	0.1566	0.1256	0.1959	0.2596
5	0.1825	0.3650	0.5400	0.0603	0.1030	0.1426	0.1164	0.1682	0.2135
6	0.1800	0.3600	0.5500	0.0647	0.1102	0.1546	0.1332	0.1963	0.2489
7	0.3000	0.4300	0.5900	0.0483	0.0650	0.0850	0.2109	0.2782	0.3560
8	0.3000	0.4300	0.5900	0.0528	0.0714	0.0927	0.2435	0.3224	0.4126
9	0.1000	0.1700	0.3400	0.0194	0.0274	0.0473	0.0870	0.1195	0.1967
10	0.0150	0.0250	0.0500	0.0066	0.0097	0.0177	0.0153	0.0202	0.0327
11	0.0150	0.0250	0.0500	0.0069	0.0106	0.0194	0.0179	0.0262	0.0449
12	0.4150	0.8300	1.6600	0.0552	0.0963	0.1843	0.2246	0.3655	0.6821
13	0.4150	0.8300	1.6600	0.0565	0.1005	0.1943	0.2285	0.3846	0.7276
14	0.4150	0.8300	1.6600	0.0519	0.0980	0.1876	0.1991	0.3715	0.7029
15	0.2250	0.4500	0.6800	0.1058	0.2048	0.3060	0.1169	0.1900	0.2651
16	0.2250	0.4500	0.6800	0.1080	0.2097	0.3135	0.1156	0.1918	0.2668
17	0.2250	0.4500	0.6800	0.1056	0.2025	0.3026	0.1252	0.1927	0.2690
18	0.2250	0.4500	0.6800	0.1067	0.2061	0.3081	0.1252	0.2020	0.2855
19	0.0125	0.0250	0.0500	0.0094	0.0164	0.0284	0.0260	0.0408	0.0639
20	0.0125	0.0250	0.0500	0.0090	0.0154	0.0273	0.0277	0.0396	0.0569
21	0.0450	0.0900	0.1800	0.0368	0.0640	0.1187	0.1088	0.1572	0.2516
22	0.1250	0.2500	0.5000	0.0814	0.1458	0.2755	0.2081	0.3086	0.5155
23	0.1250	0.2500	0.5000	0.0784	0.1450	0.2770	0.1705	0.2673	0.4527

So, let me just change the format right. So, this is the l values right for the 54 processes, this is m value, h value, this is cl, cm, ch, cl, cm, ch right. This is il, im, and ih right.

(Refer Slide Time: 32:05)

	1	2	3	4	5	6	7	8
1	12.50	25.00	50.00	9.40	16.40	28.40	26.00	40.80
2	12.50	25.00	50.00	9.00	15.40	27.30	27.70	39.60
3	45.00	90.00	180.00	36.80	64.00	118.70	108.80	157.20
4	125.00	250.00	500.00	81.40	145.80	275.50	208.10	308.60
5	125.00	250.00	500.00	78.40	145.00	277.00	170.50	267.30
6	Column 9							
7								
8								
9								131.60
10								132.40
11								134.10
12								132.00
13								104.30
14								153.20
15								95.10
16								231.50
17								207.00
18								698.70
19								537.10
20								289.20
21								657.70
22								263.10
23								672.70
24								287.40
								156.30

Similarly, you can look at the other values the raw material 1, raw material 2; it is number of processes would be 54.

(Refer Slide Time: 32:11)

```

22 - Revenue = zeros(nProcess,1); % Initialization of variable to store t
23 - penalty_domain = zeros(nProcess,1); % Initialization of variable to store t
24
25 - for j = 1: nProcess
26
27 - if x(j) >= l(j) && x(j) <= m(j) % if the production is in between l
28
29 - PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determini
30 - IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determini
31 - R1Reqd(j) = x(j)*rml(j); % Determining the raw material 1 re
32 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
33 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
34
35 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36
37 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38 - IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determini
39 - R1Reqd(j) = x(j)*rml(j); % Determining the raw material 1 r
40 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 r
41 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
42
43 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than
44
45 - penalty_domain(i) = 10^5; % Assigning penalty for violating t

```

```

14 Budget = 1000;
R>> nProcess
nProcess =
    54.00
R>> l(1)
ans =
    70.00
R>> m(1)
m(1)
↑
Error: Invalid expression. When
calling a function or indexing a
variable, use parentheses. Otherwise,
check for mismatched delimiters.

```

So, in this case number of processes is 54 right. The value of budget that we are taking is 1000. So, the production plan that we design should have an investment cost which is either thousand or less than 1000. It should not be greater than 1000. Similarly the raw material quantity which is available is 500, 500 right. So, this is just assignment. Similarly from line 18 to line 23, we are just assigning variables right.

So, PC would be used to store the production cost, IC would be used to store the investment cost of each process, R 1 required will be used to store the raw material 1 required, R 2 required is for storing the amount of raw material 2 which is required for each process. Revenue is used to store the revenue earned by selling the products from each of the process and this penalty domain will be assigned the value of 10 power 5 for those process which violet the domain constraint right.

(Refer Slide Time: 33:05)

```
32 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 req
33 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
34
35 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36
37 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38 - IC(j) = ((lh(j) - lm(j))/(h(j) - m(j)))*(x(j) - m(j)) + lm(j); % Determini
39 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 r
40 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 r
41 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
42
43 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than
44 - I
45 - penalty_domain(j) = 10^5; % Assigning penalty for violating th
46 - end
47 -
48 -
49 - end
50
51 - TotalInv8ed = sum(IC);
52 - penalty_IC = 0;
53 - if TotalInv8ed > Budget % Checking for violation of the in
54 - penalty_IC = (TotalInv8ed - Budget)^2; % Determining the penalty due to v
55 -
```

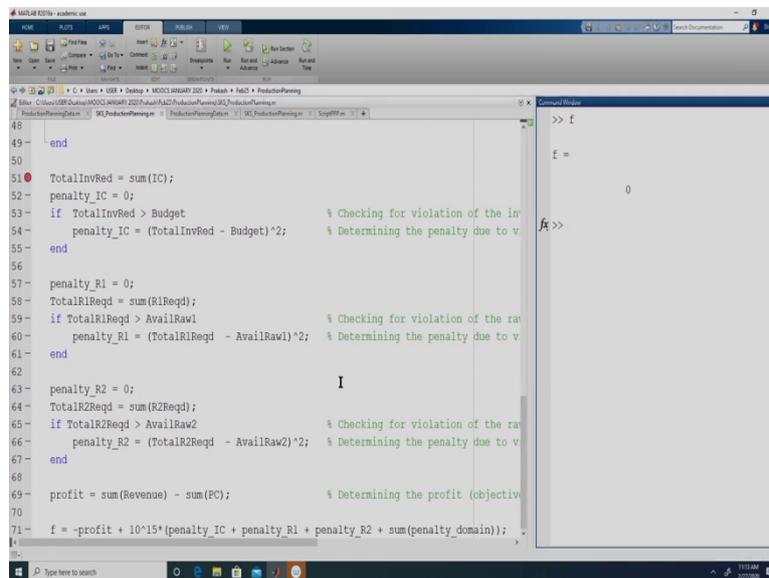
Command Window

```
ans =
    70.00
m(1)
↑
Error: Invalid expression. When
calling a function or indexing a
variable, use parentheses. Otherwise,
check for mismatched delimiters.
ans =
    135.00
h(1)
ans =
    270.00
```

Till line 23 we are just initializing the variables right. So, now, we are running this loop for each value in x right. Since all the values are 0, it will not go into this loop right. So, 1 of 1 is 1 of 1 is 70; m of 1 is 135 and h of 1 is 270 right. So, what we currently have is 0. So, 0 will not fall in this range it will not fall, it in this range and it will not even fall in this range right. So, what we are doing is if it is 0, we are not calculating the production cost investment cost or revenue or the raw materials. So, this is for the first process. So, for the second process again it will not go in any of this condition right so, because it is 0. So, even for third process, it will not go into any of this conditions right.

So, let me just put a breakpoint over here and continue. So, what it has done now it has calculated the production cost, the investment cost, the raw materials required and the revenue for all the 54 processes; let me just clear this right.

(Refer Slide Time: 34:15)



```
49 - end
50
51 TotalInvReqd = sum(IC);
52 penalty_IC = 0;
53 if TotalInvReqd > Budget % Checking for violation of the in
54     penalty_IC = (TotalInvReqd - Budget)^2; % Determining the penalty due to v
55 end
56
57 penalty_R1 = 0;
58 TotalR1Reqd = sum(R1Reqd);
59 if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw
60     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to v
61 end
62
63 penalty_R2 = 0;
64 TotalR2Reqd = sum(R2Reqd);
65 if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw
66     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to v
67 end
68
69 profit = sum(Revenue) - sum(PC); % Determining the profit (objective
70
71 f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain));
```

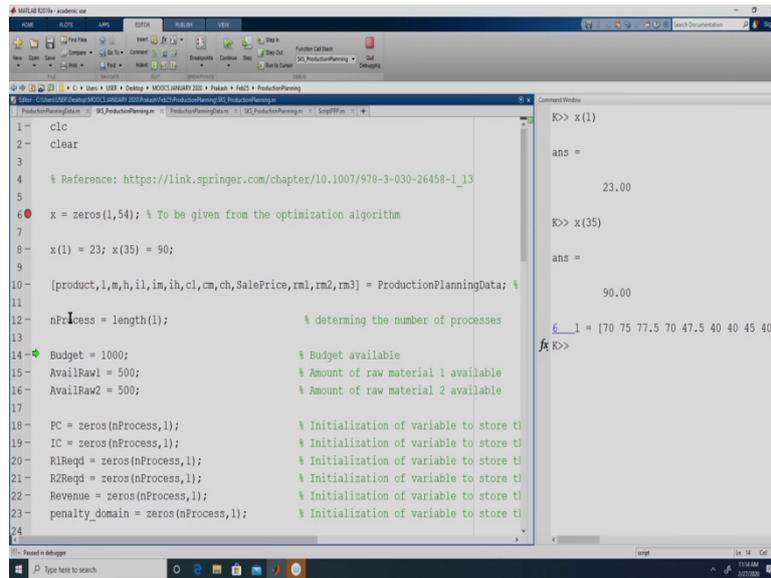
So, now we are calculating the total investment that is required right. So, IC is currently it will have a value of 0s because we initialized with 0s over here right and we did not assign any value to it right. So, this is going to be 0 right. So, penalty we are initializing it to be 0.

So, there are budget which is available to assess 1000 and what we require is only 0. So, this condition 0 is not greater than 1000. So, it will not go into this if condition right. Similarly penalty underscore R 1, we are assigning it to be 0 initially right. The raw material that we require is again 0 right and the available amount of raw material is 500. So, it again does not go into this condition. So, penalty underscore R 2 is the same thing right since we have all the variables is 0; it is not getting into this condition. So, profit would be again 0 right.

So, revenue is 0 for all the processes. So, some of that will be also 0. Similarly production cost we did not calculate the production cost at anywhere right. We initialized it to 0, we did not

calculate because it did not fall in any of this ranges right. So, this profit will be 0 right and the fitness function value would also be 0. So, f is 0 right. So, at least with heroes, it is working fine right.

(Refer Slide Time: 35:46)



```
1- clc
2- clear
3
4 % Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
5
6 x = zeros(1,54); % To be given from the optimization algorithm
7
8 x(1) = 23; x(35) = 90;
9
10 [product,l,m,h,il,im,ih,c1,cm,ch,SalePrice,rm1,rm2,rm3] = ProductionPlanningData; %
11
12 nProcess = length(l); % determining the number of processes
13
14 Budget = 1000; % Budget available
15 AvailRaw1 = 500; % Amount of raw material 1 available
16 AvailRaw2 = 500; % Amount of raw material 2 available
17
18 PC = zeros(nProcess,1); % Initialization of variable to store t
19 IC = zeros(nProcess,1); % Initialization of variable to store t
20 R1Reqd = zeros(nProcess,1); % Initialization of variable to store t
21 R2Reqd = zeros(nProcess,1); % Initialization of variable to store t
22 Revenue = zeros(nProcess,1); % Initialization of variable to store t
23 penalty_domain = zeros(nProcess,1); % Initialization of variable to store t
24
25
```

Command Window

```
E>> x(1)
ans =
    23.00
E>> x(35)
ans =
    90.00
f_x = [70 75 77.5 70 47.5 40 40 45 40 8
```

So, now what we will do it we will again do it right, but this time what we are doing is we are not giving an vector which is full of 0s. We are giving x of 1 is equal to 23 and x of 34 is equal to 90.

So, the first element is 23 and then 2 to 34, we have 0s; 35 we have 90 and then after 90, we again have set of 0s right. So, here in this case it will at least go into one of those if conditions. Let us see what happens in this case right. So, initially x is 0s right. Now we are over writing x of 1 and x of 35 right. So, if we say x of 1 is 23 x of 34 is 90 right. So, now, we are calling the data right. So, this we can just step out right.

So, all the data has been brought into the workspace right. So, the number of process is 54. These 3 lines we discussed right and again this is just initialization right.

(Refer Slide Time: 36:42)

The screenshot shows a MATLAB script in the Editor and its execution in the Command Window. The script defines variables for raw material requirements, revenue, and inventory costs based on production levels. It includes conditional logic for production constraints and a penalty function for violations. The Command Window shows the results of running the script for x=1, resulting in a revenue of 70.00 and a penalty of 23.00.

```

32 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
33 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
34
35 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36
37 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38 - IC(j) = ((lh(j) - lm(j))/(h(j) - m(j)))*(x(j) - m(j)) + lm(j); % Determini
39 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 re
40 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
41 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
42
43 - elseif 0 < x(j) && x(j) < 1(j) % if the production is greater than
44 -     penalty_domain(j) = 10^5; % Assigning penalty for violating t
45
46 - end
47 -
48 - end
49 -
50 -
51 - TotalInvRed = sum(IC);
52 - penalty_IC = 0;
53 - if TotalInvRed > Budget
54 -     penalty_IC = (TotalInvRed - Budget)^2; % Checking for violation of the im

```

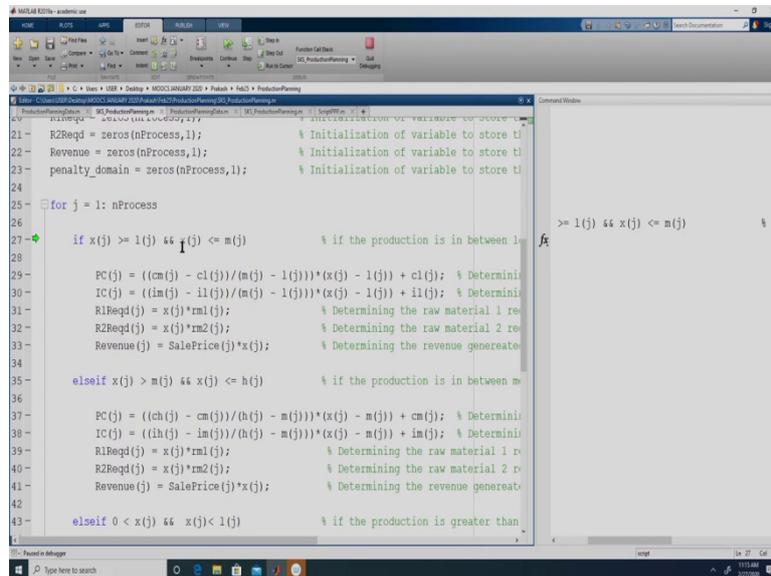
```

E>> 1(1)
ans =
    70.00
E>> x(1)
ans =
    23.00
45     penalty_domain(j) = 10^5;
47     end
49_end
27     if x(j) >= 1(j) && x(j) <= m(j)
35     elseif x(j) > m(j) && x(j) <= h(j)
43     elseif 0 < x(j) && x(j) < 1(j)
49_end
27     if x(j) >= 1(j) && x(j) <= m(j)
35     elseif x(j) > m(j) && x(j) <= h(j)
43     elseif 0 < x(j) && x(j) < 1(j)
E>>

```

So, before going to this loop just let us look at what is 1 of 1. So, 1 of 1 is 70 right and what we have is 23 right. So, if you think about it x of 1 is not 0 right. So, it is supposed to be greater than 70, but it is not greater than 70 right.

(Refer Slide Time: 36:56)

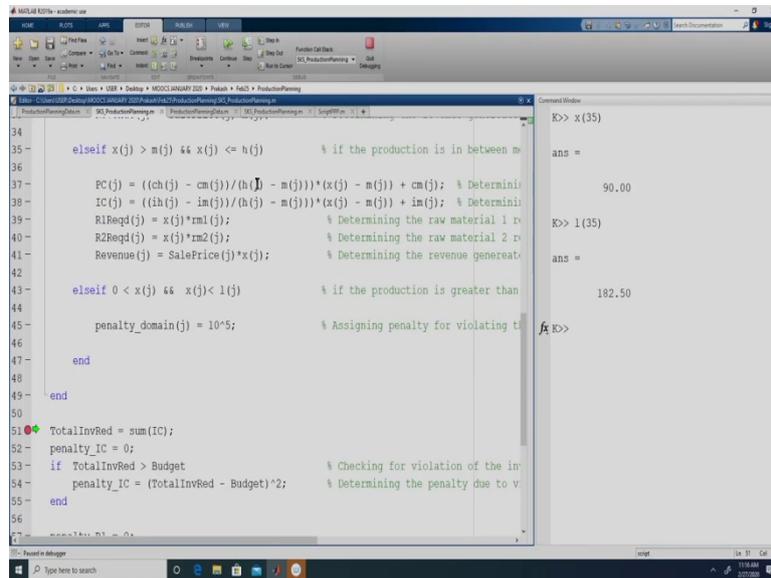


```
21- R2Reqd = zeros(nProcess,1); % Initialization of variable to store t
22- Revenue = zeros(nProcess,1); % Initialization of variable to store t
23- penalty_domain = zeros(nProcess,1); % Initialization of variable to store t
24-
25- for j = 1: nProcess
26-
27-     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between l
28-
29-         PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determini
30-         IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determini
31-         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 re
32-         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
33-         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
34-
35-     elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36-
37-         PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38-         IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determini
39-         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 re
40-         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
41-         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
42-
43-     elseif 0 < x(j) && x(j) < l(j) % if the production is greater than
```

So, this condition is not satisfied right. So, 1 of j the first value is 70. So, x of 1 is not greater than 70. So, it will not get into this part right again, it will not get into this part. So, we get into this right x of 1 is violating the domain constraint right. So, that is why it goes into this. So, this condition will be satisfied. So, we are supposed to assign a penalty for this variable right that is what we are doing over here. So, penalty domain of the first variable is 10 power 5.

So, similarly we need to calculate for all the variables right. So, for 2 again it will not go into any of these conditions right, for 3 it will not go into any of this condition right.

(Refer Slide Time: 37:37)



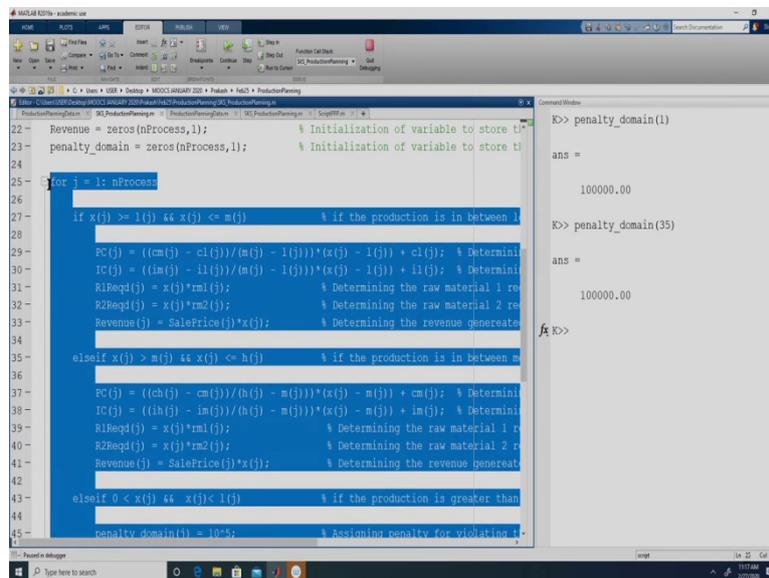
```
34
35 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36
37 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38 - IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determini
39 - R1Reqd(j) = x(j)*rml(j); % Determining the raw material 1 r
40 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 r
41 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generat
42
43 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than
44
45 - penalty_domain(j) = 10^5; % Assigning penalty for violating th
46
47 - end
48
49 - end
50
51 - TotalInvRed = sum(IC);
52 - penalty_IC = 0;
53 - if TotalInvRed > Budget % Checking for violation of the in
54 - penalty_IC = (TotalInvRed - Budget)^2; % Determining the penalty due to v
55 - end
56
57 - ===== D1 = 0;
```

Command Window

```
R>> x(35)
ans =
    90.00
R>> l(35)
ans =
   182.50
R>>
```

So, let say  $x$  of 35 is 90, let us see what is  $l$  of 35. So, that is 182.5 right. So, here also it incurs a domain constraint penalty right. So, what we will do is right we will just continue this right. So, we will say continue right.

(Refer Slide Time: 37:57)



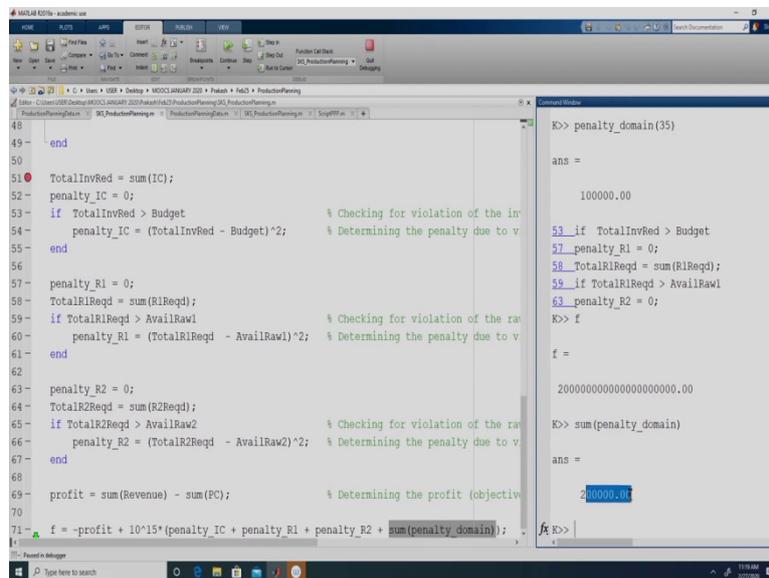
```
22 - Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue
23 - penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty domain
24
25 - for j = 1:nProcess
26
27 -     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between l and m
28
29 -         PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the production cost
30 -         IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the investment cost
31 -         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 requirement
32 -         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 requirement
33 -         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated
34
35 -     elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m and h
36
37 -         PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the production cost
38 -         IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the investment cost
39 -         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 requirement
40 -         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 requirement
41 -         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated
42
43 -     elseif 0 < x(j) && x(j) < l(j) % if the production is greater than 0 and less than l
44
45 -         penalty_domain(j) = 10^5; % Assigning penalty for violating the domain constraint
```

```
K>> penalty_domain(1)
ans =
    100000.00
K>> penalty_domain(35)
ans =
    100000.00
```

So, now let us see what is the penalty underscore domain of the first variable that is 10 power 5 and the 35th variable that is also 10 power 5 right. So, it violated 2 domain constraints.

So, 2 of the processes are active right, but both of them are violating the domain constraint right. So, we need to assign penalty for that; that is what this piece of code has helped us to do right. So, we did not calculate production cost, investment cost, revenue raw material 1, raw material 2 because the solution itself is an infeasible solution right. We cannot produce 23 from process 1 and we cannot produce 90 from process 35.

(Refer Slide Time: 38:36)



```
49 - end
50
51 TotalInvReqd = sum(IC);
52 penalty_IC = 0;
53 if TotalInvReqd > Budget % Checking for violation of the in
54     penalty_IC = (TotalInvReqd - Budget)^2; % Determining the penalty due to v
55 - end
56
57 penalty_R1 = 0;
58 TotalR1Reqd = sum(R1Reqd);
59 if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw
60     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to v
61 - end
62
63 penalty_R2 = 0;
64 TotalR2Reqd = sum(R2Reqd);
65 if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw
66     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to v
67 - end
68
69 profit = sum(Revenue) - sum(PC); % Determining the profit (objective
70
71 f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain));
```

```
K> penalty_domain(35)
ans =
    100000.00
53_if TotalInvReqd > Budget
57_penalty_R1 = 0;
58_TotalR1Reqd = sum(R1Reqd);
59_if TotalR1Reqd > AvailRaw1
63_penalty_R2 = 0;
K> f
f =
    200000000000000000000000.00
K> sum(penalty_domain)
ans =
    200000.00
fx K>
```

So, all of that would be 0 right. So, if you remember the discussion which we previously had investment cost is not calculated, production cost is not calculated, raw materials are not calculated, revenue is not calculated and profit from those processes would be 0 and since both the variables are violating in this case, the total profit is 0 right. So, that is what we are expecting. So, it did not go into this because total investment required is 0 raw material required is also 0 raw material 2 required will also be 0, the profit will also be 0 right because the variable themselves are not in the domain right.

So, there is no point of calculating profits which does not satisfy the domain constraint right. So, fitness here if we see, this profit part would be 0 other penalties are 0 right; penalty underscore IC underscore R 1 underscore R 2. These are 0 right. So, this is 10 power 5 2



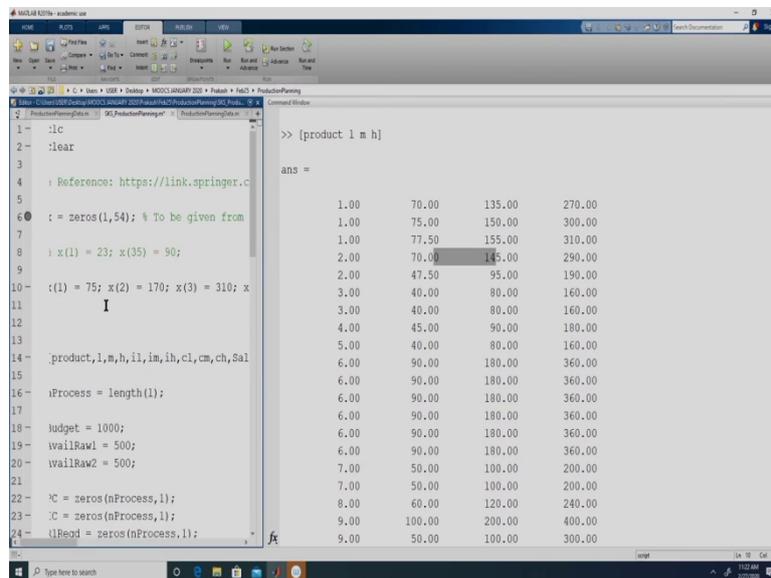
(Refer Slide Time: 40:03)

```
30 - IC(j) = ((im(j) - ll(j))/(m(j) - l(j)))*(x(j) - l(j)) + ll(j); % Determini
31 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 re
32 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 re
33 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generate
34
35 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between m
36
37 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determini
38 - IC(j) = ((lh(j) - lm(j))/(h(j) - m(j)))*(x(j) - m(j)) + lm(j); % Determini
39 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 r
40 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 r
41 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generat
42
43 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than
44
45 - penalty_domain(j) = 10^5; % Assigning penalty for violating th
46
47 - end
48
49 - end
50
51 - TotalInvRed = sum(IC);
52 - penalty_IC = 0;
53 - if TotalInvRed > Budget % Checking for violation of the inv
```

But the production which we have taken here as 23 and 90 right is infeasible.



(Refer Slide Time: 41:00)



```
1 - :lc
2 - :lear
3
4   Reference: https://link.springer.c
5
6 0 : = zeros(1,54); % To be given from
7
8 : x(1) = 23; x(35) = 90;
9
10 : (1) = 75; x(2) = 170; x(3) = 310; x
11
12
13
14 - product,1,m,h,il,im,ih,cl,cm,ch,Sal
15
16 - iProcess = length(l);
17
18 - budget = 1000;
19 - iwallRaw1 = 500;
20 - iwallRaw2 = 500;
21
22 - :C = zeros(nProcess,1);
23 - :C = zeros(nProcess,1);
24 - :lRegd = zeros(nProcess,1);
```

>> [product l m h]

ans =

1.00	70.00	135.00	270.00
1.00	75.00	150.00	300.00
1.00	77.50	155.00	310.00
2.00	70.00	145.00	290.00
2.00	47.50	95.00	190.00
3.00	40.00	80.00	160.00
3.00	40.00	80.00	160.00
4.00	45.00	90.00	180.00
5.00	40.00	80.00	160.00
6.00	90.00	180.00	360.00
6.00	90.00	180.00	360.00
6.00	90.00	180.00	360.00
6.00	90.00	180.00	360.00
6.00	90.00	180.00	360.00
6.00	90.00	180.00	360.00
7.00	50.00	100.00	200.00
7.00	50.00	100.00	200.00
8.00	60.00	120.00	240.00
9.00	100.00	200.00	400.00
9.00	50.00	100.00	300.00

So, now let us see for a feasible solution, how does it calculate. So, to give a feasible solution right let us have a look at the l m h values right. So, what we shall do is we will take process 1, 2, 3 and 4 to be active right.

(Refer Slide Time: 41:21)

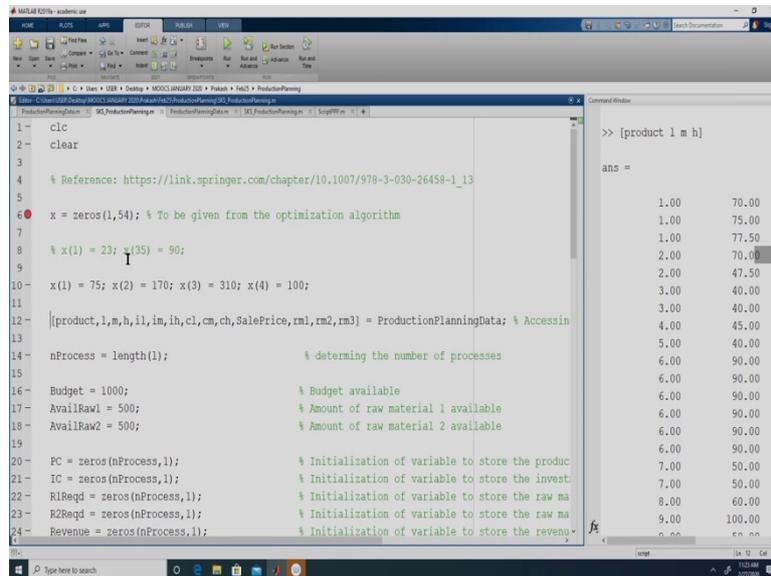
```
1 - clc
2 - clear
3
4 % Reference: https://link.springer.
5
6 x = zeros(1,54); % To be given from
7
8 x(1) = 23; x(35) = 90;
9
10 [product, l, m, h, il, im, ih, cl, cm, ch, Sa
11
12 nProcess = length(l);
13
14 Budget = 1000;
15 AvailRaw1 = 500;
16 AvailRaw2 = 500;
17
18 PC = zeros(nProcess,1);
19 IC = zeros(nProcess,1);
20 R1Reqd = zeros(nProcess,1);
21 R2Reqd = zeros(nProcess,1);
22 Revenue = zeros(nProcess,1);
23 penalty_domain = zeros(nProcess,1);
24
```

75.00	150.00	300.00
122.50	245.00	490.00
50.00	100.00	200.00
182.50	365.00	540.00
182.50	365.00	540.00
180.00	360.00	550.00
300.00	430.00	590.00
300.00	430.00	590.00
105.00	170.00	340.00
15.00	25.00	50.00
15.00	25.00	50.00
415.00	830.00	1660.00
415.00	830.00	1660.00
415.00	830.00	1660.00
225.00	450.00	680.00
225.00	450.00	680.00
225.00	450.00	680.00
225.00	450.00	680.00
12.50	25.00	50.00
12.50	25.00	50.00
45.00	90.00	180.00
125.00	250.00	500.00
125.00	250.00	500.00

So, if you remember the product vector right; so, this product vector will tell us what is the product that is being produce. So, the first 3 processes are producing product 1 and the next 2 processes are producing product 2. Let us give a feasible solution which captures all the dynamics. So, x of 1 let us say it is 75 right because it will fall in the range l m right. Let us say x of 2 is in between m and h right. So, for process 2 the m value is 150 and the h value is 300. Let us take it as 170 right. So, x of 3 let it be in this range between 150 to 310.

So, let us say it is 310 right and let us also look into one more product right. So, let us also give some value for x of 4. So, let us say x of 4 is in this range 70 to 145; 70 to 145, let us say it is 100.

(Refer Slide Time: 42:20)



```
1 - clc
2 - clear
3
4 % Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
5
6 x = zeros(1,54); % To be given from the optimization algorithm
7
8 x(1) = 23; x(35) = 90;
9
10 x(1) = 75; x(2) = 170; x(3) = 310; x(4) = 100;
11
12 [product, l, m, h, il, im, ih, cl, cm, ch, SalePrice, rml, rm2, rm3] = ProductionPlanningData; % Accessin
13
14 nProcess = length(l); % deterring the number of processes
15
16 Budget = 1000; % Budget available
17 AvailRaw1 = 500; % Amount of raw material 1 available
18 AvailRaw2 = 500; % Amount of raw material 2 available
19
20 FC = zeros(nProcess,1); % Initialization of variable to store the produc
21 IC = zeros(nProcess,1); % Initialization of variable to store the invest
22 R1Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
23 R2Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
24 Revenue = zeros(nProcess,1); % Initialization of variable to store the revenu
```

product	l	m	h	il	im	ih	cl	cm	ch	SalePrice	rml	rm2	rm3
1	1.00									70.00			
2	1.00									75.00			
3	1.00									77.50			
4	2.00									70.00			
5	2.00									47.50			
6	3.00									40.00			
7	3.00									40.00			
8	4.00									45.00			
9	5.00									40.00			
10	6.00									90.00			
11	6.00									90.00			
12	6.00									90.00			
13	6.00									90.00			
14	6.00									90.00			
15	7.00									50.00			
16	7.00									50.00			
17	8.00									60.00			
18	9.00									100.00			
19	6.00									60.00			
20	6.00									60.00			
21	6.00									60.00			
22	6.00									60.00			
23	6.00									60.00			
24	6.00									60.00			

So, you need to remember that we are not still solving the optimization problem, we are just making sure that this function which we are going to club with an meta heuristic optimization algorithm is actually calculating the values appropriately right. Once we club with meta heuristic optimization technique, we will be relying on the solution whatever we get right.

So, we are just making sure that whatever function we have written is actually calculating the values properly because the optimization algorithm is going to rely on these values. Let us execute this now right.





(Refer Slide Time: 43:12)

The screenshot shows a MATLAB script for production planning. The script defines parameters for processes, budget, raw materials, and penalties. It then iterates through processes to calculate revenue, investment, and raw material requirements based on production levels. The Command Window displays the results of these calculations for each process.

```

10 - x(1) = 75; x(2) = 170; x(3) = 310; x(4) = 100;
11
12 - [product, l, m, h, il, lm, lh, cl, cm, ch, SalePrice, rmi, rm2, rm3] = ProductionPlanningData; % Accessin
13
14 - nProcess = length(l); % determining the number of processes
15
16 - Budget = 1000; % Budget available
17 - AvailRaw1 = 500; % Amount of raw material 1 available
18 - AvailRaw2 = 500; % Amount of raw material 2 available
19
20 - PC = zeros(nProcess,1); % Initialization of variable to store the produc
21 - IC = zeros(nProcess,1); % Initialization of variable to store the invest
22 - R1Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
23 - R2Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
24 - Revenue = zeros(nProcess,1); % Initialization of variable to store the revenu
25 - penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalt
26
27 - for j = 1: nProcess
28
29 -     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and me
30 -         x: 1x54 double =
31 -             ) - 1(j)))*(x(j) - 1(j)) + cl(j); % Determining the pr
32 -             ) - 1(j)))*(x(j) - 1(j)) + il(j); % Determining the in
33 -             % Determining the raw material i required fo

```

Process	Revenue	Investment	Raw Material 1	Raw Material 2	Penalty
1	75.00	150.00	75.00	155.00	77.50
2	170.00	145.00	47.50	95.00	40.00
3	310.00	80.00	40.00	80.00	40.00
4	100.00	180.00	90.00	180.00	90.00
5	180.00	90.00	90.00	180.00	90.00
6	180.00	90.00	90.00	180.00	90.00
7	180.00	90.00	90.00	180.00	90.00
8	180.00	90.00	90.00	180.00	90.00
9	100.00	100.00	50.00	100.00	50.00
10	100.00	100.00	50.00	100.00	50.00
11	60.00	120.00	100.00	200.00	50.00
12	100.00	100.00	25.00	50.00	25.00
13	50.00	50.00	25.00	50.00	25.00
14	125.00	250.00	125.00	250.00	125.00
15	250.00	500.00	250.00	500.00	250.00

So, let us say what is l m h. So, x of 1 is 75. So, that is between l and m right. So, we expect it to go into this section.

(Refer Slide Time: 43:23)

```

25 - penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penal
26
27 - for j = 1: nProcess
28
29 -     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and me
30
31 -         PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the pr
32 -         IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the in
33 -         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required fo
34 -         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required fo
35 -         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by sell
36
37 -     elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between medium and
38
39 -         PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the pr
40 -         IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the in
41 -         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required f
42 -         R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required f
43 -         Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by sel
44
45 -     elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but
46
47 -         penalty_domain(j) = 10^5; % Assigning penalty for violating the domain
48
49 -     end
50
51 - end
52
53 - PC = PC;
54 - IC = IC;
55 - R1Reqd = R1Reqd;
56 - R2Reqd = R2Reqd;
57 - Revenue = Revenue;
58
59 - PC
60 - IC
61 - R1Reqd
62 - R2Reqd
63 - Revenue
64
65 - PC
66 - IC
67 - R1Reqd
68 - R2Reqd
69 - Revenue
70
71 - PC
72 - IC
73 - R1Reqd
74 - R2Reqd
75 - Revenue
76
77 - PC
78 - IC
79 - R1Reqd
80 - R2Reqd
81 - Revenue
82
83 - PC
84 - IC
85 - R1Reqd
86 - R2Reqd
87 - Revenue
88
89 - PC
90 - IC
91 - R1Reqd
92 - R2Reqd
93 - Revenue
94
95 - PC
96 - IC
97 - R1Reqd
98 - R2Reqd
99 - Revenue
100
101 - PC
102 - IC
103 - R1Reqd
104 - R2Reqd
105 - Revenue
106
107 - PC
108 - IC
109 - R1Reqd
110 - R2Reqd
111 - Revenue
112
113 - PC
114 - IC
115 - R1Reqd
116 - R2Reqd
117 - Revenue
118
119 - PC
120 - IC
121 - R1Reqd
122 - R2Reqd
123 - Revenue
124
125 - PC
126 - IC
127 - R1Reqd
128 - R2Reqd
129 - Revenue
130
131 - PC
132 - IC
133 - R1Reqd
134 - R2Reqd
135 - Revenue
136
137 - PC
138 - IC
139 - R1Reqd
140 - R2Reqd
141 - Revenue
142
143 - PC
144 - IC
145 - R1Reqd
146 - R2Reqd
147 - Revenue
148
149 - PC
150 - IC
151 - R1Reqd
152 - R2Reqd
153 - Revenue
154
155 - PC
156 - IC
157 - R1Reqd
158 - R2Reqd
159 - Revenue
160
161 - PC
162 - IC
163 - R1Reqd
164 - R2Reqd
165 - Revenue
166
167 - PC
168 - IC
169 - R1Reqd
170 - R2Reqd
171 - Revenue
172
173 - PC
174 - IC
175 - R1Reqd
176 - R2Reqd
177 - Revenue
178
179 - PC
180 - IC
181 - R1Reqd
182 - R2Reqd
183 - Revenue
184
185 - PC
186 - IC
187 - R1Reqd
188 - R2Reqd
189 - Revenue
190
191 - PC
192 - IC
193 - R1Reqd
194 - R2Reqd
195 - Revenue
196
197 - PC
198 - IC
199 - R1Reqd
200 - R2Reqd
201 - Revenue
202
203 - PC
204 - IC
205 - R1Reqd
206 - R2Reqd
207 - Revenue
208
209 - PC
210 - IC
211 - R1Reqd
212 - R2Reqd
213 - Revenue
214
215 - PC
216 - IC
217 - R1Reqd
218 - R2Reqd
219 - Revenue
220
221 - PC
222 - IC
223 - R1Reqd
224 - R2Reqd
225 - Revenue
226
227 - PC
228 - IC
229 - R1Reqd
230 - R2Reqd
231 - Revenue
232
233 - PC
234 - IC
235 - R1Reqd
236 - R2Reqd
237 - Revenue
238
239 - PC
240 - IC
241 - R1Reqd
242 - R2Reqd
243 - Revenue
244
245 - PC
246 - IC
247 - R1Reqd
248 - R2Reqd
249 - Revenue
250
251 - PC
252 - IC
253 - R1Reqd
254 - R2Reqd
255 - Revenue
256
257 - PC
258 - IC
259 - R1Reqd
260 - R2Reqd
261 - Revenue
262
263 - PC
264 - IC
265 - R1Reqd
266 - R2Reqd
267 - Revenue
268
269 - PC
270 - IC
271 - R1Reqd
272 - R2Reqd
273 - Revenue
274
275 - PC
276 - IC
277 - R1Reqd
278 - R2Reqd
279 - Revenue
280
281 - PC
282 - IC
283 - R1Reqd
284 - R2Reqd
285 - Revenue
286
287 - PC
288 - IC
289 - R1Reqd
290 - R2Reqd
291 - Revenue
292
293 - PC
294 - IC
295 - R1Reqd
296 - R2Reqd
297 - Revenue
298
299 - PC
300 - IC
301 - R1Reqd
302 - R2Reqd
303 - Revenue
304
305 - PC
306 - IC
307 - R1Reqd
308 - R2Reqd
309 - Revenue
310
311 - PC
312 - IC
313 - R1Reqd
314 - R2Reqd
315 - Revenue
316
317 - PC
318 - IC
319 - R1Reqd
320 - R2Reqd
321 - Revenue
322
323 - PC
324 - IC
325 - R1Reqd
326 - R2Reqd
327 - Revenue
328
329 - PC
330 - IC
331 - R1Reqd
332 - R2Reqd
333 - Revenue
334
335 - PC
336 - IC
337 - R1Reqd
338 - R2Reqd
339 - Revenue
340
341 - PC
342 - IC
343 - R1Reqd
344 - R2Reqd
345 - Revenue
346
347 - PC
348 - IC
349 - R1Reqd
350 - R2Reqd
351 - Revenue
352
353 - PC
354 - IC
355 - R1Reqd
356 - R2Reqd
357 - Revenue
358
359 - PC
360 - IC
361 - R1Reqd
362 - R2Reqd
363 - Revenue
364
365 - PC
366 - IC
367 - R1Reqd
368 - R2Reqd
369 - Revenue
370
371 - PC
372 - IC
373 - R1Reqd
374 - R2Reqd
375 - Revenue
376
377 - PC
378 - IC
379 - R1Reqd
380 - R2Reqd
381 - Revenue
382
383 - PC
384 - IC
385 - R1Reqd
386 - R2Reqd
387 - Revenue
388
389 - PC
390 - IC
391 - R1Reqd
392 - R2Reqd
393 - Revenue
394
395 - PC
396 - IC
397 - R1Reqd
398 - R2Reqd
399 - Revenue
400
401 - PC
402 - IC
403 - R1Reqd
404 - R2Reqd
405 - Revenue
406
407 - PC
408 - IC
409 - R1Reqd
410 - R2Reqd
411 - Revenue
412
413 - PC
414 - IC
415 - R1Reqd
416 - R2Reqd
417 - Revenue
418
419 - PC
420 - IC
421 - R1Reqd
422 - R2Reqd
423 - Revenue
424
425 - PC
426 - IC
427 - R1Reqd
428 - R2Reqd
429 - Revenue
430
431 - PC
432 - IC
433 - R1Reqd
434 - R2Reqd
435 - Revenue
436
437 - PC
438 - IC
439 - R1Reqd
440 - R2Reqd
441 - Revenue
442
443 - PC
444 - IC
445 - R1Reqd
446 - R2Reqd
447 - Revenue
448
449 - PC
450 - IC
451 - R1Reqd
452 - R2Reqd
453 - Revenue
454
455 - PC
456 - IC
457 - R1Reqd
458 - R2Reqd
459 - Revenue
460
461 - PC
462 - IC
463 - R1Reqd
464 - R2Reqd
465 - Revenue
466
467 - PC
468 - IC
469 - R1Reqd
470 - R2Reqd
471 - Revenue
472
473 - PC
474 - IC
475 - R1Reqd
476 - R2Reqd
477 - Revenue
478
479 - PC
480 - IC
481 - R1Reqd
482 - R2Reqd
483 - Revenue
484
485 - PC
486 - IC
487 - R1Reqd
488 - R2Reqd
489 - Revenue
490
491 - PC
492 - IC
493 - R1Reqd
494 - R2Reqd
495 - Revenue
496
497 - PC
498 - IC
499 - R1Reqd
500 - R2Reqd
501 - Revenue
502
503 - PC
504 - IC
505 - R1Reqd
506 - R2Reqd
507 - Revenue
508
509 - PC
510 - IC
511 - R1Reqd
512 - R2Reqd
513 - Revenue
514
515 - PC
516 - IC
517 - R1Reqd
518 - R2Reqd
519 - Revenue
520
521 - PC
522 - IC
523 - R1Reqd
524 - R2Reqd
525 - Revenue
526
527 - PC
528 - IC
529 - R1Reqd
530 - R2Reqd
531 - Revenue
532
533 - PC
534 - IC
535 - R1Reqd
536 - R2Reqd
537 - Revenue
538
539 - PC
540 - IC
541 - R1Reqd
542 - R2Reqd
543 - Revenue
544
545 - PC
546 - IC
547 - R1Reqd
548 - R2Reqd
549 - Revenue
550
551 - PC
552 - IC
553 - R1Reqd
554 - R2Reqd
555 - Revenue
556
557 - PC
558 - IC
559 - R1Reqd
560 - R2Reqd
561 - Revenue
562
563 - PC
564 - IC
565 - R1Reqd
566 - R2Reqd
567 - Revenue
568
569 - PC
570 - IC
571 - R1Reqd
572 - R2Reqd
573 - Revenue
574
575 - PC
576 - IC
577 - R1Reqd
578 - R2Reqd
579 - Revenue
580
581 - PC
582 - IC
583 - R1Reqd
584 - R2Reqd
585 - Revenue
586
587 - PC
588 - IC
589 - R1Reqd
590 - R2Reqd
591 - Revenue
592
593 - PC
594 - IC
595 - R1Reqd
596 - R2Reqd
597 - Revenue
598
599 - PC
600 - IC
601 - R1Reqd
602 - R2Reqd
603 - Revenue
604
605 - PC
606 - IC
607 - R1Reqd
608 - R2Reqd
609 - Revenue
610
611 - PC
612 - IC
613 - R1Reqd
614 - R2Reqd
615 - Revenue
616
617 - PC
618 - IC
619 - R1Reqd
620 - R2Reqd
621 - Revenue
622
623 - PC
624 - IC
625 - R1Reqd
626 - R2Reqd
627 - Revenue
628
629 - PC
630 - IC
631 - R1Reqd
632 - R2Reqd
633 - Revenue
634
635 - PC
636 - IC
637 - R1Reqd
638 - R2Reqd
639 - Revenue
640
641 - PC
642 - IC
643 - R1Reqd
644 - R2Reqd
645 - Revenue
646
647 - PC
648 - IC
649 - R1Reqd
650 - R2Reqd
651 - Revenue
652
653 - PC
654 - IC
655 - R1Reqd
656 - R2Reqd
657 - Revenue
658
659 - PC
660 - IC
661 - R1Reqd
662 - R2Reqd
663 - Revenue
664
665 - PC
666 - IC
667 - R1Reqd
668 - R2Reqd
669 - Revenue
670
671 - PC
672 - IC
673 - R1Reqd
674 - R2Reqd
675 - Revenue
676
677 - PC
678 - IC
679 - R1Reqd
680 - R2Reqd
681 - Revenue
682
683 - PC
684 - IC
685 - R1Reqd
686 - R2Reqd
687 - Revenue
688
689 - PC
690 - IC
691 - R1Reqd
692 - R2Reqd
693 - Revenue
694
695 - PC
696 - IC
697 - R1Reqd
698 - R2Reqd
699 - Revenue
700
701 - PC
702 - IC
703 - R1Reqd
704 - R2Reqd
705 - Revenue
706
707 - PC
708 - IC
709 - R1Reqd
710 - R2Reqd
711 - Revenue
712
713 - PC
714 - IC
715 - R1Reqd
716 - R2Reqd
717 - Revenue
718
719 - PC
720 - IC
721 - R1Reqd
722 - R2Reqd
723 - Revenue
724
725 - PC
726 - IC
727 - R1Reqd
728 - R2Reqd
729 - Revenue
730
731 - PC
732 - IC
733 - R1Reqd
734 - R2Reqd
735 - Revenue
736
737 - PC
738 - IC
739 - R1Reqd
740 - R2Reqd
741 - Revenue
742
743 - PC
744 - IC
745 - R1Reqd
746 - R2Reqd
747 - Revenue
748
749 - PC
750 - IC
751 - R1Reqd
752 - R2Reqd
753 - Revenue
754
755 - PC
756 - IC
757 - R1Reqd
758 - R2Reqd
759 - Revenue
760
761 - PC
762 - IC
763 - R1Reqd
764 - R2Reqd
765 - Revenue
766
767 - PC
768 - IC
769 - R1Reqd
770 - R2Reqd
771 - Revenue
772
773 - PC
774 - IC
775 - R1Reqd
776 - R2Reqd
777 - Revenue
778
779 - PC
780 - IC
781 - R1Reqd
782 - R2Reqd
783 - Revenue
784
785 - PC
786 - IC
787 - R1Reqd
788 - R2Reqd
789 - Revenue
790
791 - PC
792 - IC
793 - R1Reqd
794 - R2Reqd
795 - Revenue
796
797 - PC
798 - IC
799 - R1Reqd
800 - R2Reqd
801 - Revenue
802
803 - PC
804 - IC
805 - R1Reqd
806 - R2Reqd
807 - Revenue
808
809 - PC
810 - IC
811 - R1Reqd
812 - R2Reqd
813 - Revenue
814
815 - PC
816 - IC
817 - R1Reqd
818 - R2Reqd
819 - Revenue
820
821 - PC
822 - IC
823 - R1Reqd
824 - R2Reqd
825 - Revenue
826
827 - PC
828 - IC
829 - R1Reqd
830 - R2Reqd
831 - Revenue
832
833 - PC
834 - IC
835 - R1Reqd
836 - R2Reqd
837 - Revenue
838
839 - PC
840 - IC
841 - R1Reqd
842 - R2Reqd
843 - Revenue
844
845 - PC
846 - IC
847 - R1Reqd
848 - R2Reqd
849 - Revenue
850
851 - PC
852 - IC
853 - R1Reqd
854 - R2Reqd
855 - Revenue
856
857 - PC
858 - IC
859 - R1Reqd
860 - R2Reqd
861 - Revenue
862
863 - PC
864 - IC
865 - R1Reqd
866 - R2Reqd
867 - Revenue
868
869 - PC
870 - IC
871 - R1Reqd
872 - R2Reqd
873 - Revenue
874
875 - PC
876 - IC
877 - R1Reqd
878 - R2Reqd
879 - Revenue
880
881 - PC
882 - IC
883 - R1Reqd
884 - R2Reqd
885 - Revenue
886
887 - PC
888 - IC
889 - R1Reqd
890 - R2Reqd
891 - Revenue
892
893 - PC
894 - IC
895 - R1Reqd
896 - R2Reqd
897 - Revenue
898
899 - PC
900 - IC
901 - R1Reqd
902 - R2Reqd
903 - Revenue
904
905 - PC
906 - IC
907 - R1Reqd
908 - R2Reqd
909 - Revenue
910
911 - PC
912 - IC
913 - R1Reqd
914 - R2Reqd
915 - Revenue
916
917 - PC
918 - IC
919 - R1Reqd
920 - R2Reqd
921 - Revenue
922
923 - PC
924 - IC
925 - R1Reqd
926 - R2Reqd
927 - Revenue
928
929 - PC
930 - IC
931 - R1Reqd
932 - R2Reqd
933 - Revenue
934
935 - PC
936 - IC
937 - R1Reqd
938 - R2Reqd
939 - Revenue
940
941 - PC
942 - IC
943 - R1Reqd
944 - R2Reqd
945 - Revenue
946
947 - PC
948 - IC
949 - R1Reqd
950 - R2Reqd
951 - Revenue
952
953 - PC
954 - IC
955 - R1Reqd
956 - R2Reqd
957 - Revenue
958
959 - PC
960 - IC
961 - R1Reqd
962 - R2Reqd
963 - Revenue
964
965 - PC
966 - IC
967 - R1Reqd
968 - R2Reqd
969 - Revenue
970
971 - PC
972 - IC
973 - R1Reqd
974 - R2Reqd
975 - Revenue
976
977 - PC
978 - IC
979 - R1Reqd
980 - R2Reqd
981 - Revenue
982
983 - PC
984 - IC
985 - R1Reqd
986 - R2Reqd
987 - Revenue
988
989 - PC
990 - IC
991 - R1Reqd
992 - R2Reqd
993 - Revenue
994
995 - PC
996 - IC
997 - R1Reqd
998 - R2Reqd
999 - Revenue
1000 - PC
1001 - IC
1002 - R1Reqd
1003 - R2Reqd
1004 - Revenue
1005 - PC
1006 - IC
1007 - R1Reqd
1008 - R2Reqd
1009 - Revenue
1010 - PC
1011 - IC
1012 - R1Reqd
1013 - R2Reqd
1014 - Revenue
1015 - PC
1016 - IC
1017 - R1Reqd
1018 - R2Reqd
1019 - Revenue
1020 - PC
1021 - IC
1022 - R1Reqd
1023 - R2Reqd
1024 - Revenue
1025 - PC
1026 - IC
1027 - R1Reqd
1028 - R2Reqd
1029 - Revenue
1030 - PC
1031 - IC
1032 - R1Reqd
1033 - R2Reqd
1034 - Revenue
1035 - PC
1036 - IC
1037 - R1Reqd
1038 - R2Reqd
1039 - Revenue
1040 - PC
1041 - IC
1042 - R1Reqd
1043 - R2Reqd
1044 - Revenue
1045 - PC
1046 - IC
1047 - R1Reqd
1048 - R2Reqd
1049 - Revenue
1050 - PC
1051 - IC
1052 - R1Reqd
1053 - R2Reqd
1054 - Revenue
1055 - PC
1056 - IC
1057 - R1Reqd
1058 - R2Reqd
1059 - Revenue
1060 - PC
1061 - IC
1062 - R1Reqd
1063 - R2Reqd
1064 - Revenue
1065 - PC
1066 - IC
1067 - R1Reqd
1068 - R2Reqd
1069 - Revenue
1070 - PC
1071 - IC
1072 - R1Reqd
1073 - R2Reqd
1074 - Revenue
1075 - PC
1076 - IC
1077 - R1Reqd
1078 - R2Reqd
1079 - Revenue
1080 - PC
1081 - IC
1082 - R1Reqd
1083 - R2Reqd
1084 - Revenue
1085 - PC
1086 - IC
1087 - R1Reqd
1088 - R2Reqd
1089 - Revenue
1090 - PC
1091 - IC
1092 - R1Reqd
1093 - R2Reqd
1094 - Revenue
1095 - PC
1096 - IC
1097 - R1Reqd
1098 - R2Reqd
1099 - Revenue
1100 - PC
1101 - IC
1102 - R1Reqd
1103 - R2Reqd
1104 - Revenue
1105 - PC
1106 - IC
1107 - R1Reqd
1108 - R2Reqd
1109 - Revenue
1110 - PC
1111 - IC
1112 - R1Reqd
1113 - R2Reqd
1114 - Revenue
1115 - PC
1116 - IC
1117 - R1Reqd
1118 - R2Reqd
1119 - Revenue
1120 - PC
1121 - IC
1122 - R1Reqd
1123 - R2Reqd
1124 - Revenue
1125 - PC
1126 - IC
1127 - R1Reqd
1128 - R2Reqd
1129 - Revenue
1130 - PC
1131 - IC
1132 - R1Reqd
1133 - R2Reqd
1134 - Revenue
1135 - PC
1136 - IC
1137 - R1Reqd
1138 - R2Reqd
1139 - Revenue
1140 - PC
1141 - IC
1142 - R1Reqd
1143 - R2Reqd
1144 - Revenue
1145 - PC
1146 - IC
1147 - R1Reqd
1148 - R2Reqd
1149 - Revenue
1150 - PC
1151 - IC
1152 - R1Reqd
1153 - R2Reqd
1154 - Revenue
1155 - PC
1156 - IC
1157 - R1Reqd
1158 - R2Reqd
1159 - Revenue
1160 - PC
1161 - IC
1162 - R1Reqd
1163 - R2Reqd
1164 - Revenue
1165 - PC
1166 - IC
1167 - R1Reqd
1168 - R2Reqd
1169 - Revenue
1170 - PC
1171 - IC
1172 - R1Reqd
1173 - R2Reqd
1174 - Revenue
1175 - PC
1176 - IC
1177 - R1Reqd
1178 - R2Reqd
1179 - Revenue
1180 - PC
1181 - IC
1182 - R1Reqd
1183 - R2Reqd
1184 - Revenue
1185 - PC
1186 - IC
1187 - R1Reqd
1188 - R2Reqd
1189 - Revenue
1190 - PC
1191 - IC
1192 - R1Reqd
1193 - R2Reqd
1194 - Revenue
1195 - PC
1196 - IC
1197 - R1Reqd
1198 - R2Reqd
1199 - Revenue
1200 - PC
1201 - IC
1202 - R1Reqd
1203 - R2Reqd
1204 - Revenue
1205 - PC
1206 - IC
1207 - R1Reqd
1208 - R2Reqd
1209 - Revenue
1210 - PC
1211 - IC
1212 - R1Reqd
1213 - R2Reqd
1214 - Revenue
1215 - PC
1216 - IC
1217 - R1Reqd
1218 - R2Reqd
1219 - Revenue
1220 - PC
1221 - IC
1222 - R1Reqd
1223 - R2Reqd
1224 - Revenue
1225 - PC
1226 - IC
1227 - R1Reqd
1228 - R2Reqd
1229 - Revenue
1230 - PC
1231 - IC
1232 - R1Reqd
1233 - R2Reqd
1234 - Revenue
1235 - PC
1236 - IC
1237 - R1Reqd
1238 - R2Reqd
1239 - Revenue
1240 - PC
1241 - IC
1242 - R1Reqd
1243 - R2Reqd
1244 - Revenue
1245 - PC
1246 - IC
1247 - R1Reqd
1248 - R2Reqd
1249 - Revenue
1250 - PC
1251 - IC
1252 - R1Reqd
1253 - R2Reqd
1254 - Revenue
1255 - PC
1256 - IC
1257 - R1Reqd
1258 - R2Reqd
1259 - Revenue
1260 - PC
1261 - IC
1262 - R1Reqd
1263 - R2Reqd
1264 - Revenue
1265 - PC
1266 - IC
1267 - R1Reqd
1268 - R2Reqd
1269 - Revenue
1270 - PC
1271 - IC
1272 - R1Reqd
1273 - R2Reqd
1274 - Revenue
1275 - PC
1276 - IC
1277 - R1Reqd
1278 - R2Reqd
1279 - Revenue
1280 - PC
1281 - IC
1282 - R1Reqd
1283 - R2Reqd
1284 - Revenue
1285 - PC
1286 - IC
1287 - R1Reqd
1288 - R2Reqd
1289 - Revenue
1290 - PC
1291 - IC
1292 - R1Reqd
1293 - R2Reqd
1294 - Revenue
1295 - PC
1296 - IC
1297 - R1Reqd
1298 - R2Reqd
1299 - Revenue
1300 - PC
1301 - IC
1302 - R1Reqd
1303 - R2Reqd
1304 - Revenue
1305 - PC
1306 - IC
1307 - R1Reqd
1308 - R2Reqd
1309 - Revenue
1310 - PC
1311 - IC
1312 - R1Reqd
1313 - R2Reqd
1314 - Revenue
1315 - PC
1316 - IC
1317 - R1Reqd
1318 - R2Reqd
1319 - Revenue
1320 - PC
1321 - IC
1322 - R1Reqd
1323 - R2Reqd
1324 - Revenue
1325 - PC
1326 - IC
1327 - R1Reqd
1328 - R2Reqd
1329 - Revenue
1330 - PC
1331 - IC
1332 - R1Reqd
1333 - R2Reqd
1334 - Revenue
1335 - PC
1336 - IC
1337 - R1Reqd
1338 - R2Reqd
1339 - Revenue
1340 - PC
1341 - IC
1342 - R1Reqd
1343 - R2Reqd
1344 - Revenue
1345 - PC
1346 - IC
1347 - R1Reqd
1348 - R2Reqd
1349 - Revenue
1350 - PC
1351 - IC
1352 - R1Reqd
1353 - R2Reqd
1354 - Revenue
1355 - PC
1356 - IC
1357 - R1Reqd
1358 - R2Reqd
1359 - Revenue
1360 - PC
1361 - IC
1362 - R1Reqd
1363 - R2Reqd
1364 - Revenue
1365 - PC
1366 - IC
1367 - R1Reqd
1368 - R2Reqd
1369 - Revenue
1370 - PC
1371 - IC
1372 - R1Reqd
1373 - R2Reqd
1374 - Revenue
1375 - PC
1376 - IC
1377 - R1Reqd
1378 - R2Reqd
1379 - Revenue
1380 - PC
1381 - IC
1382 - R1Reqd
1383 - R2Reqd
1384 - Revenue
1385 - PC
1386 - IC
1387 - R1Reqd
1388 - R2Reqd
1389 - Revenue
1390 - PC
1391 - IC
1392 - R1Reqd
1393 - R2Reqd
1394 - Revenue
1395 - PC
1396 - IC
1397 - R1Reqd
1398 - R2Reqd
1399 - Revenue
1400 - PC
1401 - IC
1402 - R1Reqd
1403 - R2Reqd
1404 - Revenue
1405 - PC
1406 - IC
1407 - R1Reqd
1408 - R2Reqd
1409 - Revenue
1410 - PC
1411 - IC
1412 - R1Reqd
1413 - R2Reqd
1414 - Revenue
1415 - PC
1416 - IC
1417 - R1Reqd
1418 - R2Reqd
1419 - Revenue
1420 - PC
1421 - IC
1422 - R1Reqd
1423 - R2Reqd
1424 - Revenue
1425 - PC
1426 - IC
1427 - R1Reqd
1428 - R2Reqd
1429 - Revenue
1430 - PC
1431 - IC
1432 - R1Reqd
1433 - R2Reqd
1434 - Revenue
1435 - PC
1436 - IC
1437 - R1Reqd
1438 - R2Reqd
1439 - Revenue
1440 - PC
1441 - IC
1442 - R1Reqd
1443 - R2Reqd
1444 - Revenue
1445 - PC
1446 - IC
1447 - R1Reqd
1448 - R2Reqd
1449 - Revenue
1450 - PC
1451 - IC
1452 - R1Reqd
1453 - R2Reqd
1454 - Revenue
1455 - PC
1456 - IC
1457 - R1Reqd
1458 - R2Reqd
1459 - Revenue
1460 - PC
1461 - IC
1462 - R1Reqd
1463 - R2Reqd
1464 - Revenue
1465 - PC
1466 - IC
1467 - R1Reqd
1468 - R2Reqd
1469 - Revenue
1470 - PC
1471 - IC
1472 - R1Reqd
1473 - R2Reqd
1474 - Revenue
1475 - PC
1476 - IC
1477 - R1Reqd
1478 - R2Reqd
1479 - Revenue
1480 - PC
1481 - IC
1482 - R1Reqd
1483 - R2Reqd
1484 - Revenue
1485 - PC
1486 - IC
1487 - R1Reqd
1488 - R2Reqd
1489 - Revenue
1490 - PC
1491 - IC
1492 - R1Reqd
1493 - R2Reqd
1494 - Revenue
1495 - PC
1496 - IC
1497 - R1Reqd
1498 - R2Reqd
1499 - Revenue
1500 - PC
1501 - IC
1502 - R1Reqd
1503 - R2Reqd
1504 - Revenue
1505 - PC
1506 - IC
1507 - R1Reqd
1508 - R2Reqd
1509 - Revenue
1510 - PC
1511 - IC
1512 - R1Reqd
1513 - R2Reqd
1514 - Revenue
1515 - PC
1516 - IC
1517 - R1Reqd
1518 - R2Reqd
1519 - Revenue
1520 - PC
1521 - IC
1522 - R1Reqd
1523 - R2Reqd
1524 - Revenue
1525 - PC
1526 - IC
1527 - R1Reqd
1528 - R2Reqd
1529 - Revenue
1530 - PC
1531 - IC
1532 - R1Reqd
1533 - R2Reqd
1534 - Revenue
1535 - PC
1536 - IC
1537 - R1Reqd
1538 - R2Reqd
1539 - Revenue
1540 - PC
1541 - IC
1542 - R1Reqd
1543 - R2Reqd
1544 - Revenue
1545 - PC
1546 - IC
1547 - R1Reqd
1548 - R2Reqd
1549 - Revenue
1550 - PC
1551 - IC
1552 - R1Reqd
1553 - R2Reqd
1554 - Revenue
1555 - PC
1556 - IC
1557 - R1Reqd
1558 - R2Reqd
1559 - Revenue
1560 - PC
1561 - IC
1562 - R1Reqd
1563 - R2Reqd
1564 - Revenue
1565 - PC
1566 - IC
1567 - R1Reqd
1568 - R2Reqd
1569 - Revenue
1570 - PC
1571 - IC
1572 - R1Reqd
1573 - R2Reqd
1574 - Revenue
1575 - PC
1576 - IC
1577 - R1Reqd
1578 - R2Reqd
1579 - Revenue
1580 - PC
1581 - IC
1582 - R1Reqd
1583 - R2Reqd
1584 - Revenue
1585 - PC
1586 - IC
1587 - R1Reqd
1588 - R2Reqd
1589 - Revenue
1590 - PC
1591 - IC
1592 - R1Reqd
1593 - R2Reqd
1594 - Revenue
1595 - PC
1596 - IC
1597 - R1Reqd
1598 - R2Reqd
1599 - Revenue
1600 - PC
1601 - IC
1602 - R1Reqd
1603 - R2Reqd
1604 - Revenue
1605 - PC
1606 - IC
1607 - R1Reqd
1608 - R2Reqd
1609 - Revenue
1610 - PC
1611 - IC
1612 - R1Reqd
1613 - R2Reqd
1614 - Revenue
1615 - PC
1616 - IC
1617 - R1Reqd
1618 - R2Reqd
1619 - Revenue
1620 - PC
1621 - IC
1622 - R1Reqd
1623 - R2Reqd
1624 - Revenue
1625 - PC
1626 - IC
1627 - R1Reqd
1628 - R2Reqd
1629 - Revenue
1630 - PC
1631 - IC
1632 - R1Reqd
1633 - R2Reqd
1634 - Revenue
1635 - PC
1636 - IC
1637 - R1Reqd
1638 - R2Reqd
1639 - Revenue
1640 - PC
1641 - IC
1642 - R1Reqd
1643 - R2Reqd
1644 - Revenue
1645 - PC
1646 - IC
1647 - R1Reqd
1648 - R2Reqd
1649 - Revenue
1650 - PC
1651 - IC
1652 - R1Reqd
1653 - R2Reqd
1654 - Revenue
1655 - PC
1656 - IC
1657 - R1Reqd
1658 - R2Reqd
1659 - Revenue
166
```

(Refer Slide Time: 44:19)

```

31 - PC(j) = ((cm(j) - c1(j))/(m(j) - l(j)))*(x(j) - l(j)) + c1(j); % Determining the pr
32 - IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining the in
33 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required fo
34 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required fo
35 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by sell
36
37 - elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between medium and
38
39 - PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the pr
40 - IC(j) = ((ih(j) - im(j))/(h(j) - m(j)))*(x(j) - m(j)) + im(j); % Determining the in
41 - R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required f
42 - R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2 required f
43 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by sel
44
45 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but
46
47 - penalty_domain(j) = 10^5; % Assigning penalty for violating the domain
48
49 - end
50
51 - end
52
53 - TotalInvRed = sum(IC);

```

200.00
540.00
540.00
550.00
590.00
590.00
340.00
50.00
50.00
1660.00
1660.00
1660.00
680.00
680.00
680.00
680.00
50.00
50.00
180.00
500.00
500.00

So, process 2 if you remember, we had taken a value such that it goes into this section right. So, here again we are calculating the production cost right 116.1 2 investment cost is 91.41. So, raw material 1 that is required is 160.34, raw material 2 required is 0 the revenue earned from selling the products which come out of process 2 is 164.74 right and this we can continue for the other processes right. So, for x 3 also it goes between m and h right. So, for x 4, it was 100 right. So, it goes into this range right l of 4 is 70 right and m of 4 is 145. So, 100 is in between that.

So, production cost is 70.06, investment cost is 66.3. Similarly we calculate raw material 1 that is required. So, raw material 2 required is 0 right and the revenue from the process 4 is 97.5 right. So, since we had only 4 active processes right for the rest of the processes, it is 0.

So, it will not go into any of these conditions right. So, the production cost and the investment cost will be 0 for rest of the processes. Let us continue right.

(Refer Slide Time: 45:35)

```

50
51 - end
52
53 TotalInvRed = sum(IC);
54 penalty_IC = 0;
55 if TotalInvRed > Budget % Checking for violation of the investment
56     penalty_IC = (TotalInvRed - Budget)^2; % Determining the penalty due to violation
57 - end
58
59 penalty_R1 = 0;
60 TotalR1Reqd = sum(R1Reqd);
61 if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw materia
62     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to violation
63 - end
64
65 penalty_R2 = 0;
66 TotalR2Reqd = sum(R2Reqd);
67 if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw materia
68     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to violation
69 - end
70
71 profit = sum(Revenue) - sum(C); % Determining the profit (objective)
72
73 f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain)); % Determ

```

Variable	Value
TotalInvRed	200.00
penalty_IC	540.00
TotalR1Reqd	540.00
penalty_R1	550.00
TotalR2Reqd	590.00
penalty_R2	340.00
profit	50.00
f	50.00
TotalR1Reqd	1660.00
penalty_R1	1660.00
TotalR2Reqd	680.00
penalty_R2	680.00
profit	680.00
f	50.00
TotalR2Reqd	50.00
penalty_R2	180.00
profit	500.00
f	500.00

So, now we have calculated. So, now we have calculated for all the processes. So, the total investment cost is 348 right which is nothing, but the summation of these 4 values. So, penalty we are initializing as 0 right. So, budget what we had was 1000 right. So, what we require is 348.81 right. So, 348.81 is not greater than thousand so, no penalty right. So, that is why I did not go into this line. Similarly raw material 1 that is required is 621.09 and what we have is 500 right.

So, this production right which does not violate any of the domain constraint actually violates the raw material constraint 1 right. So, raw material 1 that is required 621 and what we have is only 400 right. So, the penalty we calculate by 621.09 minus 500 the whole square. So, that is

the penalty with respect to raw material 1 right. So, raw material 2, there will not be any penalty for this production plan because none of the 4 processes use raw material 2 right. So, the profit for this is the sum of revenue minus the sum of production cost right.

(Refer Slide Time: 46:51)

The screenshot shows a MATLAB script for production planning. The script calculates total inventory required, total raw material required for two materials, and total production cost. It also checks for violations of investment and raw material constraints, calculating penalties for each. The profit is calculated as the sum of revenue minus the sum of production cost and penalties. The command window shows the following output:

```

ans =
    180.00
    500.00
    500.00

PC(j) = ((cm(j) - c
K>> PC(1:4)
ans =
    53.73
    116.12
    195.70
    70.06

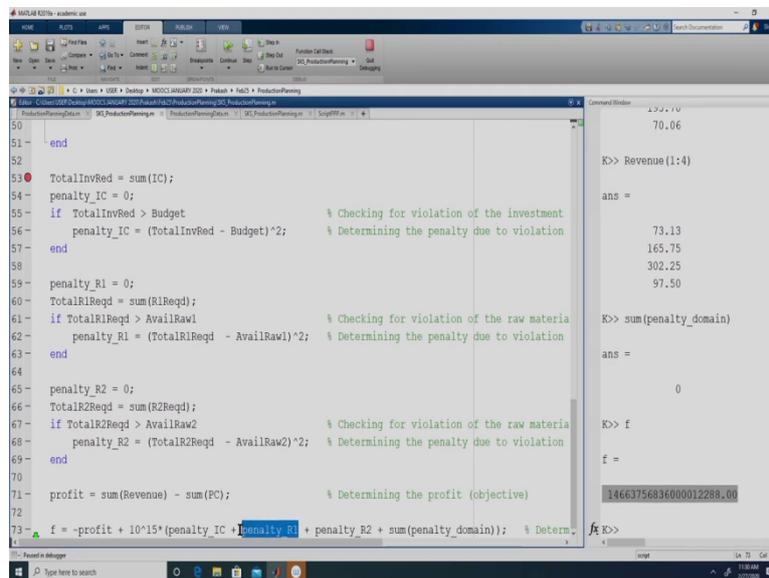
Revenue(1:4)
ans =
    73.13
    165.75
    302.25
    97.50

sum(penalty_domain)

```

So, production cost for the first 4 processes are this one and the revenue earned from these 4 processes is this one right. So, the difference between them is the profit right. So, we get a profit of 203.01 right. Though this production plan has a profit associated with it right, but we cannot implement it because it violates the raw material constraint right. So, let us see what will be the value of fitness. So, this will be 203.01, this is penalty underscore IC is 0, penalty underscore R 2 is 0 and the sum of penalty domain will also be 0 because all the 4 active processes do not violate the domain constraint right.

(Refer Slide Time: 47:31)



```
50
51 - end
52
53 TotalInvRed = sum(IC);
54 penalty_IC = 0;
55 if TotalInvRed > Budget % Checking for violation of the investment
56     penalty_IC = (TotalInvRed - Budget)^2; % Determining the penalty due to violation
57 end
58
59 penalty_R1 = 0;
60 TotalR1Reqd = sum(R1Reqd);
61 if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw materia
62     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to violation
63 end
64
65 penalty_R2 = 0;
66 TotalR2Reqd = sum(R2Reqd);
67 if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw materia
68     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to violation
69 end
70
71 profit = sum(Revenue) - sum(PC); % Determining the profit (objective)
72
73 f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain)); % Determ
```

Command Window

```
70.06
K>> Revenue(1:4)
ans =
    73.13
   165.75
   302.25
    97.50
K>> sum(penalty_domain)
ans =
     0
K>> f
f =
 14663756836000012288.00
```

So, this is also 0 right; only penalty we have is the penalty with respect to the raw material 1. So, that we are multiplying by a very high factor 10 power 15 and we are calculating the fitness for it right. So, f is this high value right. So, it is a high value because it is a infeasible solution. The infeasibility is due to the violation of the constraint related to raw material. So, that is how this fitness function file is going to work right, given a solution it is now helping us to calculate the fitness of that solution. If you carefully analyze, you will be able to understand; if it is a feasible solution, then the penalty value is going to be 0 right.

So, 10 power 15 into 0 is going to be 0 and profit would be a positive value right, but fitness is equal to minus of profit. If the fitness value which is f in this case if f is negative, we can be confident that it is a feasible solution because that will happen only when the penalty is 0 and the penalty will be 0 for a feasible solution and whatever the value of fitness function is comes from only the profit. So, for this case profit is a positive quantity and we are taking negative of

that right. So, for this case study at least right so, whenever we get a negative value we can be confident that it is a feasible solution right.

So, now that we have looked into this, now let us plug it with an optimization algorithm. To plug it with an optimization algorithm we need to convert into a function file because remember that we will be calling this function multiple times. So, if you are executing let say teaching learning based optimization, we will be calling it in during the initialization on of population, will be calling it multiple times in teacher face, we will be calling it in learner face right. So, this file which is a script file as of now needs to become a function file.

(Refer Slide Time: 49:18)

```

1 function f = SKS_ProductionPlanning(x)
2
3 [product, l, m, h, il, lm, lh, cl, cm, ch, SalePrice, rm1, rm2, rm3] = ProductionPlanningData; % Accessin
4
5 nProcess = length(l); % determining the number of processes
6
7 Budget = 1000; % Budget available
8 AvailRaw1 = 500; % Amount of raw material 1 available
9 AvailRaw2 = 500; % Amount of raw material 2 available
10
11 PC = zeros(nProcess,1); % Initialization of variable to store the produc
12 IC = zeros(nProcess,1); % Initialization of variable to store the invest
13 R1Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
14 R2Reqd = zeros(nProcess,1); % Initialization of variable to store the raw ma
15 Revenue = zeros(nProcess,1); % Initialization of variable to store the revenu
16 penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalt
17
18 for j = 1: nProcess
19     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and me
20
21         PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining the pr
22         IC(j) = ((lm(j) - ll(j))/(m(j) - l(j)))*(x(j) - l(j)) + ll(j); % Determining the in
23         R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 required fo
24

```

Command Window Output:

```

K>> Revenue(1:4)
ans =
    70.06
    73.13
    165.75
    302.25
    97.50

K>> sum(penalty_domain)
ans =
     0

K>> f
f =
14663756836000012288.00

```

So, here we have that function file right. So, what we have done is we have removed that x values right because the x values are now going to come from an algorithm and this function is merely going to return the fitness function value f right. So, what we will now do is we will

plug it with teaching learning based optimization right. So, this is the teaching learning based optimization algorithm which we had developed right. So, we are not going to modify anything in this algorithm.

(Refer Slide Time: 49:33)

The screenshot shows a MATLAB R2016a window with a script editor on the left and a command window on the right. The script defines a function `TLBO` for teaching learning based optimization. The command window shows the execution of `TLBO` with a fitness function `f` and a parameter vector `p`.

```

1 function [bestsol,bestf,tnexs,BestFitter,P,F] = TLBO(prob,lb,ub,Np,T)
2
3 %% Starting of TLBO
4 f = NaN(Np,1); % Vector to store the fitness function value of the
5 BestFitter = NaN(T+1,1); % Vector to store the best fitness function value
6
7 D = length(lb); % Determining the number of decision variables in
8
9 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial
10
11 for p = 1:Np
12     f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial p
13 end
14 BestFitter(1) = min(f);
15
16 %% Iteration loop
17 for t = 1:T
18     for i = 1:Np
19         %% Teacher Phase
20         Xmean = mean(f); % Determining mean of the population
21         [~,indl] = min(f); % Determining the location of the teacher
22     end
23
24

```

The command window shows the following output:

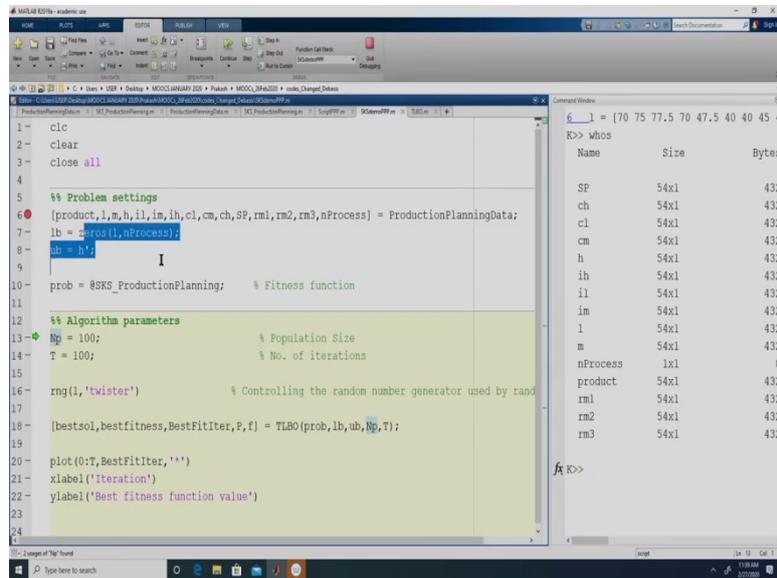
```

6 _1 = [70 75 77.5 70 47.5 40 40 45 4
K>> whos
Name      Size      Bytes
SP        54x1      432
ch        54x1      432
cl        54x1      432
cm        54x1      432
h         54x1      432
ih        54x1      432
il        54x1      432
im        54x1      432
l         54x1      432
m         54x1      432
nProcess  1x1       8
product   54x1      432
rm1       54x1      432
rm2       54x1      432
rm3       54x1      432
f
K>>

```

So, we will now solve it with teaching learning based optimization right. So, this is the code that we had developed previously for TLBO right.

(Refer Slide Time: 49:54)



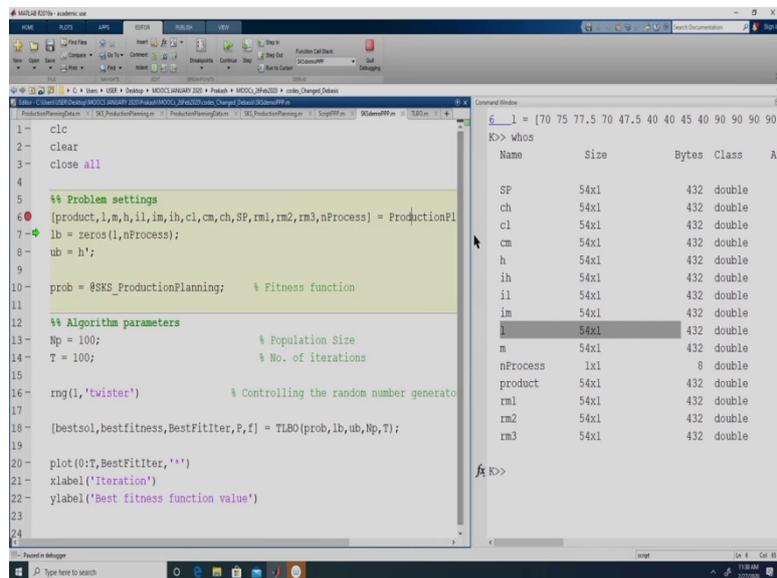
```
1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 [product,l,m,h,il,im,lb,cl,cm,ch,SP,rm1,rm2,rm3,nProcess] = ProductionPlanningData;
7 lb = zeros(1,nProcess);
8 ub = h;
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1,'twister') % Controlling the random number generator used by rand
17
18 [bestsol,bestfitness,BestFitIter,P,f] = TLBO(prob,lb,ub,Np,T);
19
20 plot(0:T,BestFitIter,')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24
```

Command Window

```
6 l = [70 75 77.5 70 47.5 40 40 45 4
K> whos
Name      Size      Bytes
SP        54x1      432
ch        54x1      432
cl        54x1      432
cm        54x1      432
h         54x1      432
ih        54x1      432
il        54x1      432
im        54x1      432
l         54x1      432
m         54x1      432
nProcess  1x1        8
product   54x1      432
rm1       54x1      432
rm2       54x1      432
rm3       54x1      432
```

So, what we will do now is we will a develop a script file right. So, we have this script file right. So, for this function right, we need to provide the lower bound upper bound; we need to tell the name of the file where the fitness function is calculated. We need to give the population size  $N_p$  and we need to give the number of iterations that is to be performed right. So, that is why we are defining over here. So, the upper bound is the  $h$  value right. So, instead of directly giving the  $h$  value we are calling it from the file production planning data right.

(Refer Slide Time: 50:20)



The screenshot shows the MATLAB Editor with a script file open. The script contains the following code:

```
1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 [product,l,m,h,il,im,ih,cl,cm,ch,SP,rm1,rm2,rm3,nProcess] = ProductionPl
7 lb = zeros(1,nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1,'twister') % Controlling the random number generator
17
18 [bestsol,bestfitness,BestFitIter,P,f] = TLBO(prob,lb,ub,Np,T);
19
20 plot(0:T,BestFitIter, '*')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24
```

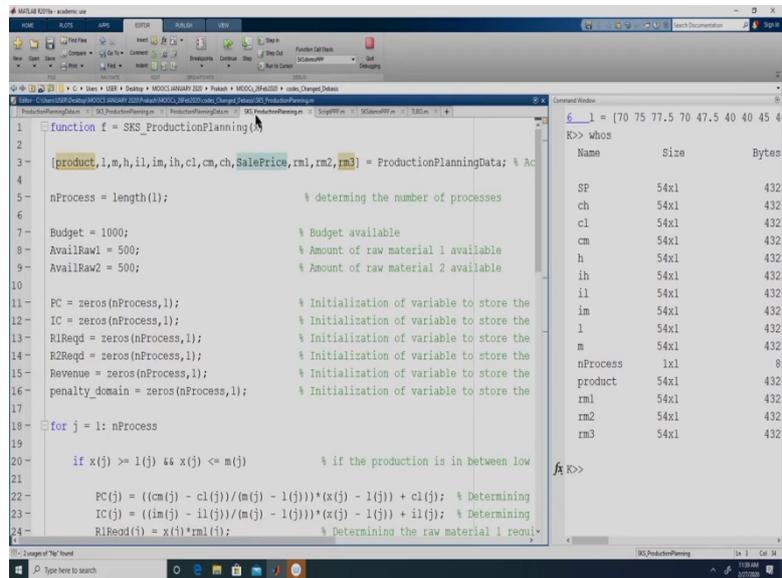
The Command Window shows the execution of the script, displaying the values of the variables defined in line 6:

```
6 l = [70 75 77.5 70 47.5 40 40 45 40 90 90 90 90]
K>> whos
Name      Size      Bytes  Class  At
SP        54x1     432    double
ch        54x1     432    double
cl        54x1     432    double
cm        54x1     432    double
h         54x1     432    double
ih        54x1     432    double
il        54x1     432    double
im        54x1     432    double
l         54x1     432    double
m         54x1     432    double
nProcess  1x1       8      double
product   54x1     432    double
rm1       54x1     432    double
rm2       54x1     432    double
rm3       54x1     432    double
```

What this line 6 will do is it will help us fetch all the data right ah. So, let me just step out of this function right. So, now, all the data which we want is over here right. So, for example, selling price production cost of low level, medium level, high level right; the investment cost related to the 3 levels l values, m values and rest of all the values are available in this script file because this was your function file. Remember this function file only contains the data right. So, we did not supply any input to this function file, we merely called this function file and it helped us with all the details related to the problem right.

So, lower bound as we discussed it is not l, but it is 0s; 0s of 1 comma n process upper bound is the h value right. So, since we needed the h value over here that is why in line 6 we had to call this function file right. So, this upper bound is available now right.

(Refer Slide Time: 51:16)



```
function f = SKS_ProductionPlanning(x)
%
[product, l, m, h, il, im, ih, cl, cm, ch, SalePrice, rm1, rm2, rm3] = ProductionPlanningData; % Ac
%
nProcess = length(l); % determining the number of processes
%
Budget = 1000; % Budget available
AvailRaw1 = 500; % Amount of raw material 1 available
AvailRaw2 = 500; % Amount of raw material 2 available
%
PC = zeros(nProcess,1); % Initialization of variable to store the
IC = zeros(nProcess,1); % Initialization of variable to store the
R1Reqd = zeros(nProcess,1); % Initialization of variable to store the
R2Reqd = zeros(nProcess,1); % Initialization of variable to store the
Revenue = zeros(nProcess,1); % Initialization of variable to store the
penalty_domain = zeros(nProcess,1); % Initialization of variable to store the
%
for j = 1: nProcess
    if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low
        PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determining
        IC(j) = ((im(j) - il(j))/(m(j) - l(j)))*(x(j) - l(j)) + il(j); % Determining
        R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1 requi
```

```
6 l = [70 75 77.5 70 47.5 40 40 45 4
K> whos
Name      Size      Bytes
SP        54x1      432
ch        54x1      432
cl        54x1      432
cm        54x1      432
h         54x1      432
ih        54x1      432
il        54x1      432
im        54x1      432
l         54x1      432
m         54x1      432
nProcess  1x1        8
product   54x1      432
rm1       54x1      432
rm2       54x1      432
rm3       54x1      432
```

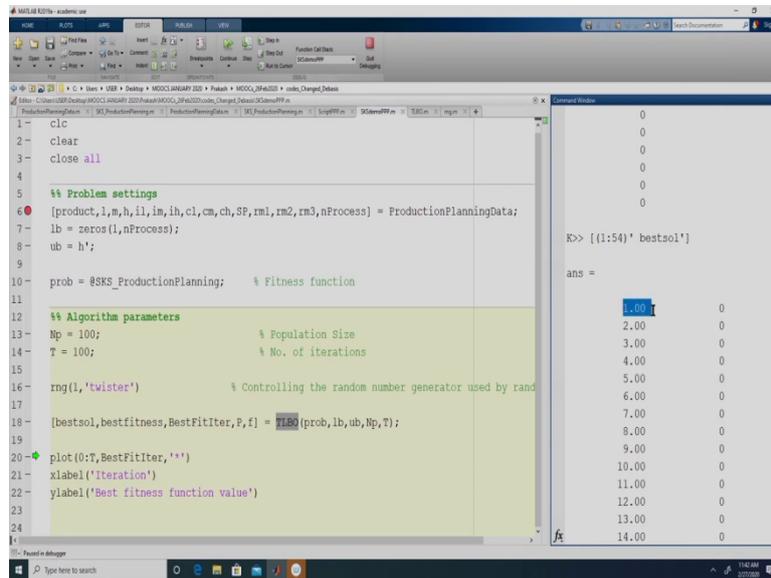
So, the problem is in this file right. So, SKS underscore ProductionPlanning that is what we have given over here. So, the lower bound is given, the upper bound is given, the file to determine the fitness function value is also given right. So, with respect to problem, we will required 3 things which have been defined right this  $N_p$  is the population size,  $T$  is the number of iterations right.

So, this rng we have discussed previously, it helps to control the random numbers right. So, that if required we can regenerate the results right. We are fixing the algorithm to be twister algorithm and we are fixing the seed to be 1. So, now, what we are doing is we are providing all the required details to the TLB algorithm; only now that we are solving it as an optimization problem right. So, what we expect from the TLBO algorithm is the best solution right, the solution as in like what are the decision variables.





(Refer Slide Time: 53:16)



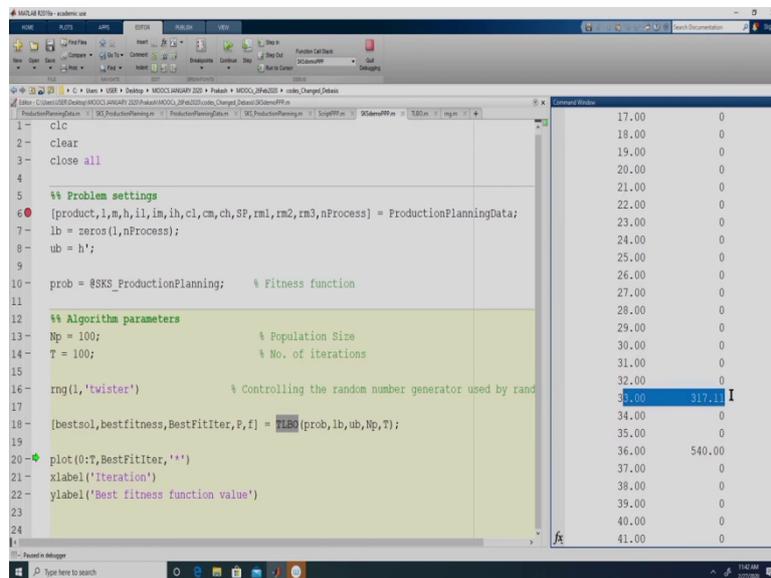
```
1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 [product,l,m,h,ll,lm,lh,cl,cm,ch,SP,rm1,rm2,rm3,nProcess] = ProductionPlanningData;
7 lb = zeros(1,nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1,'twister') % Controlling the random number generator used by rand
17
18 [bestsol,bestfitness,BestFitIter,P,F] = TLBO(prob,lb,ub,Np,T);
19
20 plot(0:T,BestFitIter, '*')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24
```

Command Window

```
0
0
0
0
0
0
K>> [(1:54)' bestsol']
ans =
1.00 0
2.00 0
3.00 0
4.00 0
5.00 0
6.00 0
7.00 0
8.00 0
9.00 0
10.00 0
11.00 0
12.00 0
13.00 0
14.00 0
```

So, here what we did was from 1 to 54 we generated a vector right, it will generate a row vector and since we want the column vector we use this transpose symbol right. Similarly bestsol what we got was a row vector. So, we are transposing it to get a column vector. Now we have appended this column. So, that we know from which process how much is to be produced right.

(Refer Slide Time: 53:35)



So, here if we see nothing is produced till process 32 right. So, process 33 we are supposed to produce 317.11, process 36 we need to produce 540 units.

(Refer Slide Time: 53:46)

```

1- clc
2- clear
3- close all
4
5 %% Problem settings
6 [product, l, m, h, il, lm, lh, cl, cm, ch, SP, rml, rm2, rm3, nProcess] = ProductionPlanningData;
7 lb = zeros(1, nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1, 'twister') % Controlling the random number generator used by rand
17
18 [bestsol, bestfitness, BestFitIter, F, f] = YLABO(prob, lb, ub, Np, T);
19
20 plot(0:T, BestFitIter, '*')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24

```

Process	Product	Best Fitness Function Value
1	32.00	0
2	33.00	317.11
3	34.00	0
4	35.00	0
5	36.00	540.00
6	37.00	0
7	38.00	0
8	39.00	0
9	40.00	0
10	41.00	0
11	42.00	0
12	43.00	0
13	44.00	724.01
14	45.00	0
15	46.00	0
16	47.00	0
17	48.00	680.00
18	49.00	0
19	50.00	0
20	51.00	0
21	52.00	0
22	53.00	0
23	54.00	0

Process 44, we need to produce 724.01, process 48 we need to produce 680 right and that is it. So, only 4 processes are active right. If you want to know what is the product als so, we can have here product right.

(Refer Slide Time: 54:07)

The image shows a MATLAB Editor window with a script on the left and a Command Window on the right. The script defines problem settings, algorithm parameters, and uses the `fmincon` function to solve a production planning problem. The Command Window displays the resulting product vector `ans`.

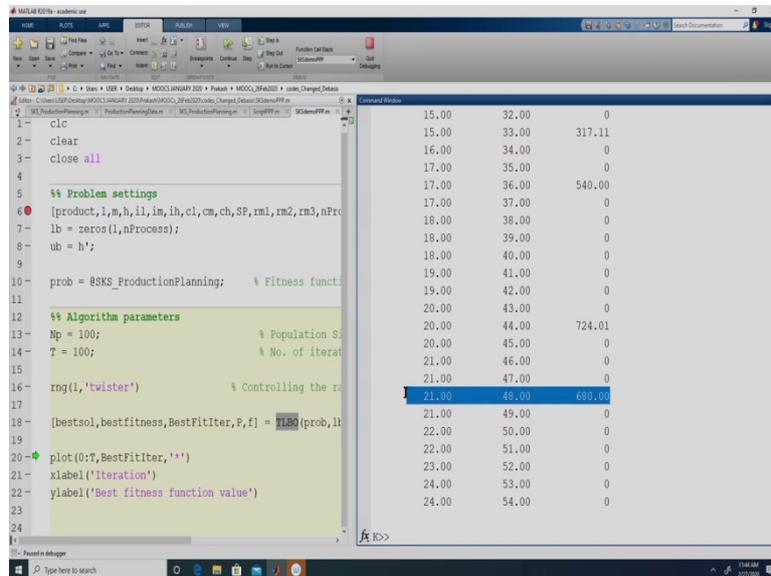
```
1- clc
2- clear
3- close all
4
5 %% Problem settings
6 [product, l, m, h, il, lm, lh, cl, cm, ch, SP, rml, rm2, rm3, nPro
7- lb = zeros(1, nProcess);
8- ub = h';
9
10- prob = @SKS_ProductionPlanning; % Fitness functi
11
12 %% Algorithm parameters
13- Np = 100; % Population S
14- T = 100; % No. of iterat
15
16- rng(1, 'twister') % Controlling the ra
17
18- [bestsol, bestfitness, BestFitIter, F, f] = fmincon(prob, lb
19
20- plot(0:T, BestFitIter, '*')
21- xlabel('Iteration')
22- ylabel('Best fitness function value')
23
24
```

The Command Window output is:

```
ans =
    1.00    1.00    0
    1.00    2.00    0
    1.00    3.00    0
    2.00    4.00    0
    2.00    5.00    0
    3.00    6.00    0
    3.00    7.00    0
    4.00    8.00    0
    5.00    9.00    0
    6.00   10.00    0
    6.00   11.00    0
    6.00   12.00    0
    6.00   13.00    0
    6.00   14.00    0
    6.00   15.00    0
    7.00   16.00    0
    7.00   17.00    0
    8.00   18.00    0
    9.00   19.00    0
   10.00  20.00    0
   10.00  22.00    0
   11.00  23.00    0
```

So, this product vector we had discussed that each value will tell what is the product that is produced. So, for example, the first 3 values are 1 so; that means, process 1, process 2, process 3 produce product 1 that is how we are assigned this variable product right.

(Refer Slide Time: 54:26)



So, now we have product in the first column right. So, let us go to the non zero values. So, this is to be interpreted as from process 33, we need to produce 317.10 and the product that is produced by process 33 is product 15 right. Similarly this is product 17 is being produced from process 36 and the production quantity is 540. Over here the product 20 is being produced using process 44.

So, product 20, we can use 43, 44, 45, but the algorithm has chosen 44 for the optimal solution. So, product 20 from process 44 the amount of production is 724.01. Similarly over here product 21 is being produced from process 48 and the amount of production is 680. So, that is how we analyze this bestsol right.

(Refer Slide Time: 55:15)

The screenshot shows a MATLAB script in the Editor window and its execution results in the Command Window. The script defines problem settings, algorithm parameters, and uses the `TLBO` function to solve the optimization problem. The Command Window displays a table of fitness values for various iterations and the final best fitness value.

```
1- clc
2- clear
3- close all
4
5 %% Problem settings
6 [product, l, m, h, il, lm, lh, cl, cm, ch, SP, rml, rm2, rm3, nProcess] = ProductionPlannin
7 lb = zeros(1, nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1, 'twister') % Controlling the random number generator use
17
18 [bestsol, bestfitness, BestFitIter, F, f] = TLBO(prob, lb, ub, Np, T);
19
20 plot(0:T, BestFitIter, '*')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24
```

Iteration	Best fitness function value
18.00	38.00
18.00	39.00
18.00	40.00
19.00	41.00
19.00	42.00
20.00	43.00
20.00	44.00
20.00	44.00
20.00	45.00
21.00	46.00
21.00	47.00
21.00	48.00
21.00	49.00
22.00	50.00
22.00	51.00
23.00	52.00
24.00	53.00
24.00	54.00

Command Window Output:

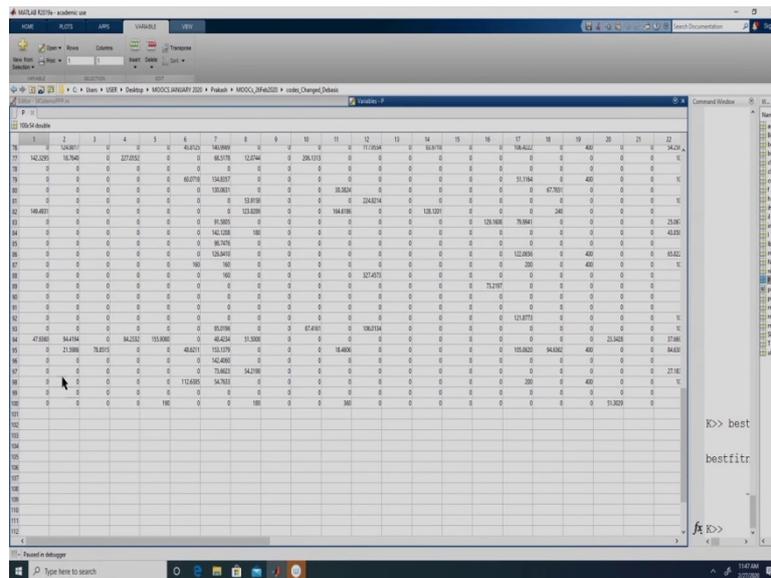
```
E>> bestfitness
bestfitness =
-413.80
```

So, let us look at what is best fitness. Best fitness is the fitness function associated with this solution right. So, it is negative we had discussed previously at least for this case study with this data a negative value will indicate a feasible solution. So, this is a feasible solution what we have is a feasible solution. So, if you go back and plug this solution into this file SKS underscore ProductionPlanning, you can calculate what is the raw material that is required what is the investment cost that is required.

So, all that post optimality analysis can be done right. So, for this solution the fitness value is minus 413.80. For the current case this will happen only when the penalty is 0. So, the profit that we have is 413.80. Remember what the algorithm has solved is a minimization problem and what we had is a maximization problem right.

So, whatever value we get from the algorithm, we need to multiply it by a minus sign right. So, the actual profit is 413.80. Though it shows minus 413, it shows it with respect to the fitness, but the profit is 413.80 right. We will come to this BestFitIter a little bit later right, let us look at what is the population what is P right. So, P is the population in the last iteration.

(Refer Slide Time: 56:37)



So, since it is going to have 54 values, let us look it in the workspace later. So, the P is over here. So, this we will have 100 rows right because each row indicates the population number and each column indicates the variable. So, we will have 54 columns right.

(Refer Slide Time: 56:52)

The screenshot displays a Microsoft Excel spreadsheet with a grid of data. The columns are labeled from 30 to 55, and the rows are numbered from 1 to 17. The data consists of numerical values, with many cells containing zero. The values are organized in a structured manner, likely representing a matrix or a set of parameters. On the right side of the spreadsheet, a command window is visible, showing the text 'E>> best' and 'bestfitr', indicating a search or optimization process. The Excel interface includes standard menus like File, Home, Insert, and Formulas, and a status bar at the bottom showing the current cell address and values.

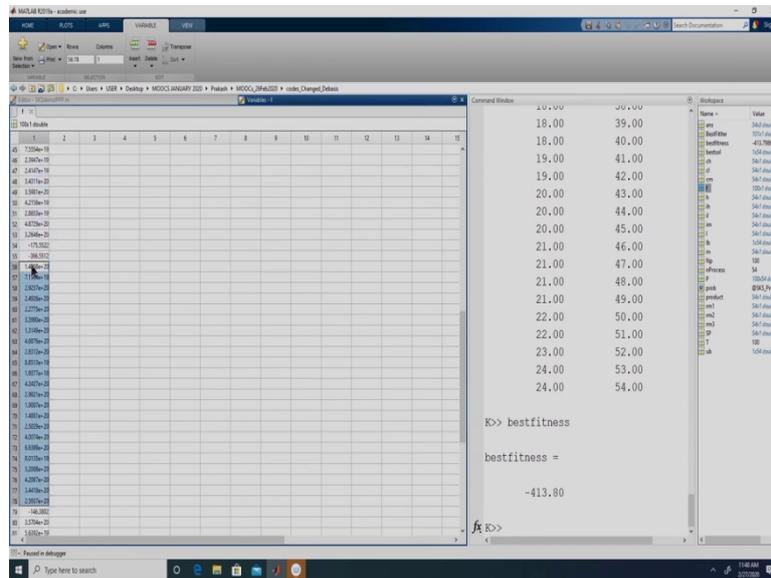
So, if we keep going we should have value till 54 right. So, each of this is a solution right.

(Refer Slide Time: 56:57)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	73.9471	0	0	204.7491	171.8237	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

So, you can see that lot of these values have become 0. So, 0 means from that process, we are not producing anything right. So, the fitness function corresponding to each of this solution. So, these are the solution; not only the optimal solution the algorithm also gives us 99 other solutions right. So, if you want to look at the fitness of this 100 solution, it will be in the vector f. Let us look into the vector f over here right.

(Refer Slide Time: 57:21)



So, here if we see many of these solutions are still infeasible right. All of these are infeasible only those which have a negative sign are feasible right. So, this is the fitness function corresponding to each of the 100 solutions.

(Refer Slide Time: 57:40)

```

1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 [product, l, m, h, ll, lm, lh, cl, cm, ch, SP, rml, rm2, rm3, nProcess] = ProductionP
7 lb = zeros(1, nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1, 'twister') % Controlling the random number generator
17
18 [bestsol, bestfitness, BestFitIter, P, f] = TLBO(prob, lb, ub, Np, T);
19
20 plot(0:T, BestFitIter, 'r')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
23
24

```

Command Window:

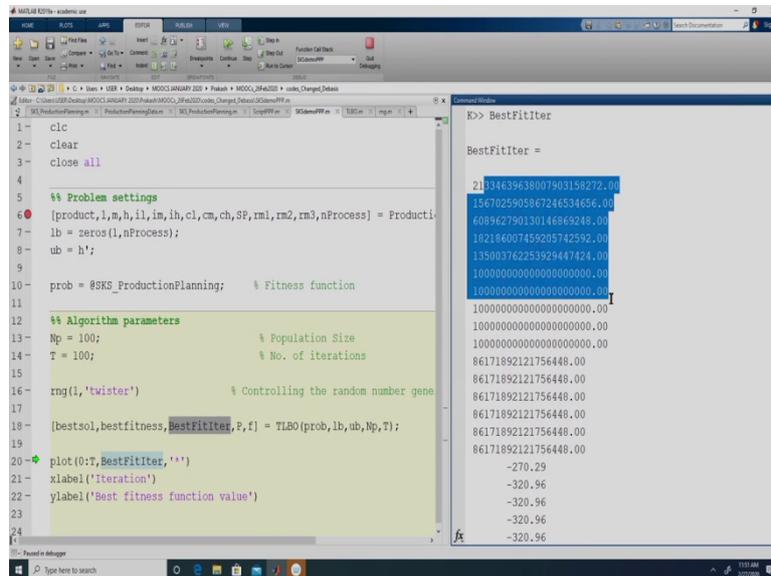
```

ans =
    413.80
   -374.47
   -366.55
   -355.14
   -275.90
   -261.43
   -196.00
   -175.55
   -146.38
1193603437619421440.00
1937721024382550272.00
3447654454810629632.00
1123600000000000000.00
14419907495869976576.00
16301752595999113216.00
2394749664022233088.00
24147295236000002048.00
28652795057028493312.00
3461606406458833056.00
39282456757672566784.00
42156435290896220160.00
43980029875588882432.00
44182005664444444440.00

```

As a small post optimality analysis what we can do is we can say sort of  $f$  right. So, this is  $f$  that has been sorted right; so, out of 100 solutions 3 4 5 6 7 8 9. So, even after 100 iterations right out of the 100 population number, we have only 9 feasible solution. So, the rest of the solutions are infeasible solutions that is the meaning of this  $P$  and  $f$  right.

(Refer Slide Time: 58:05)

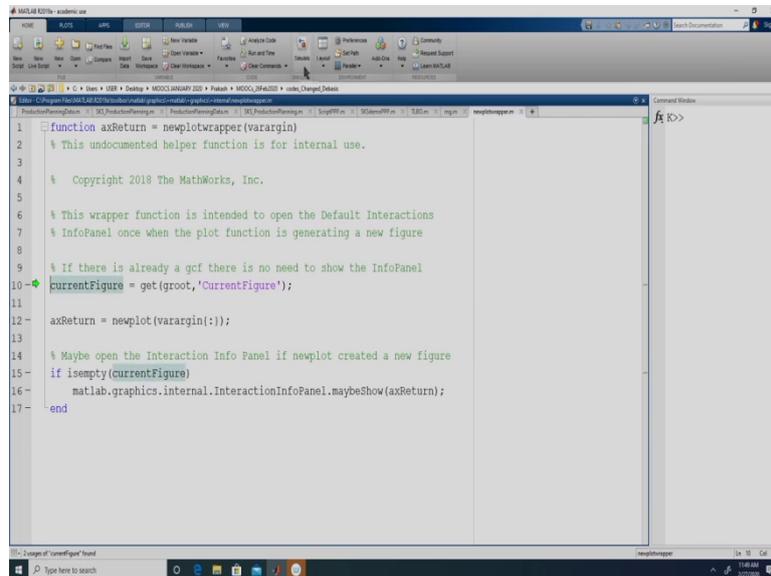


```
1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 [product,l,m,h,il,lm,lh,c1,cm,ch,SP,rm1,rm2,rm3,nProcess] = Producti
7 lb = zeros(1,nProcess);
8 ub = h';
9
10 prob = @SKS_ProductionPlanning; % Fitness function
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 rng(1,'twister') % Controlling the random number gene
17
18 [bestsol,bestfitness,BestFitIter,P,f] = TLBO(prob,lb,ub,Np,T);
19
20 plot(0:T,BestFitIter,')
21 xlabel('Iteration')
22 ylabel('Best fitness function value')
```

```
K>> BestFitIter
BestFitIter =
21 134639638007903158272.40
1567025905867246534636.00
608962790130146869248.00
182186007459205742592.00
13500376225392447424.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
1000000000000000000.00
86171892121756448.00
86171892121756448.00
86171892121756448.00
86171892121756448.00
86171892121756448.00
86171892121756448.00
-270.29
-320.96
-320.96
-320.96
-320.96
```

So, now let us look at BestFitIter right. So, BestFitIter we can actually plot right because if you remember the previous discussion that is the value for convergence curve right. So, what we are you doing over here is we are plotting on the x axis 0 to T; 0 to T because BestFitIter will have 101 values though our number of iterations are 100 because it will also store what is the best value in the initial population right

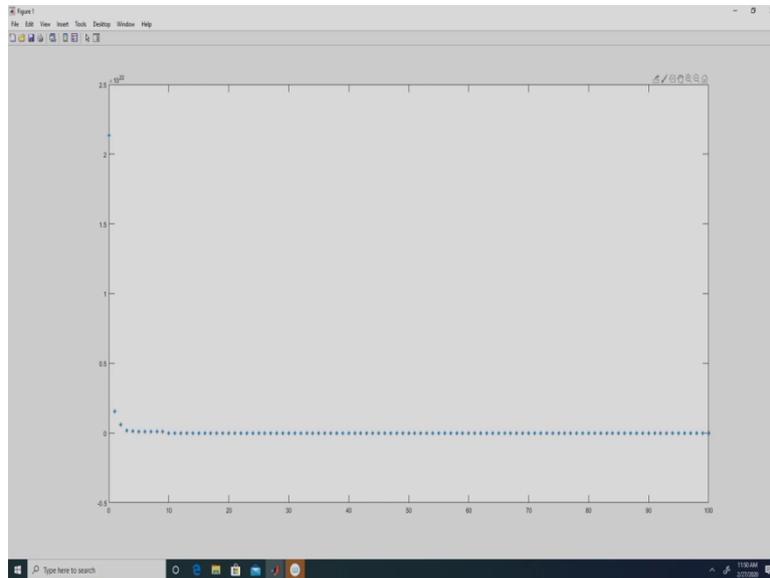
(Refer Slide Time: 58:32)



```
1 function axReturn = newplotwrapper(varargin)
2 % This undocumented helper function is for internal use.
3
4 % Copyright 2018 The MathWorks, Inc.
5
6 % This wrapper function is intended to open the Default Interactions
7 % InfoPanel once when the plot function is generating a new figure
8
9 % If there is already a(gcf) there is no need to show the InfoPanel
10 currentFigure = get(gcf, 'CurrentFigure');
11
12 axReturn = newplot(varargin{:});
13
14 % Maybe open the Interaction Info Panel if newplot created a new figure
15 if isempty(currentFigure)
16     matlab.graphics.internal.InteractionInfoPanel.maybeShow(axReturn);
17 end
```

So, let us have a look at that figure right.

(Refer Slide Time: 58:35)

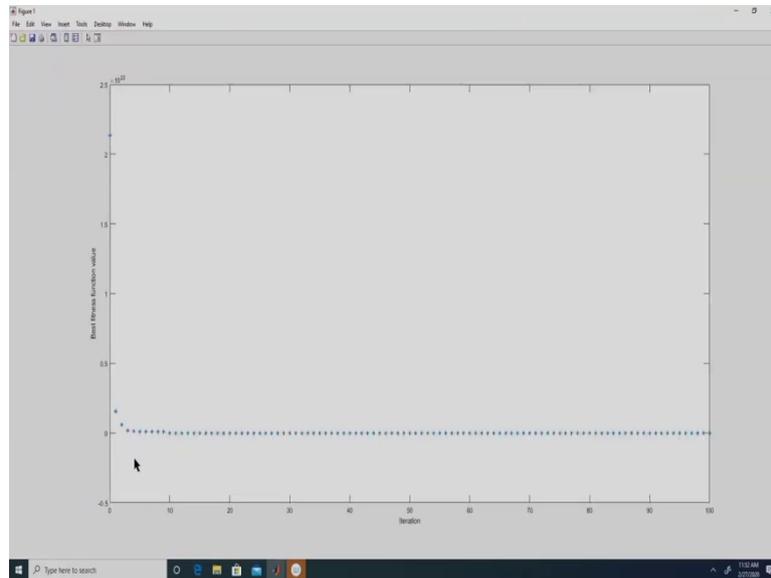


So, this is the convergence curve. So, if you look at this convergence curve, it might seem like nothing much is happening after let us say 11th or 12th iteration, but that is because this magnitude is very high  $10^{22}$  right. So, if you look at this BestFitIter values right so, initial population had a fitness of this one. The best member in the initial population had a fitness of this value right and as iteration progressed the solution started to improve right.



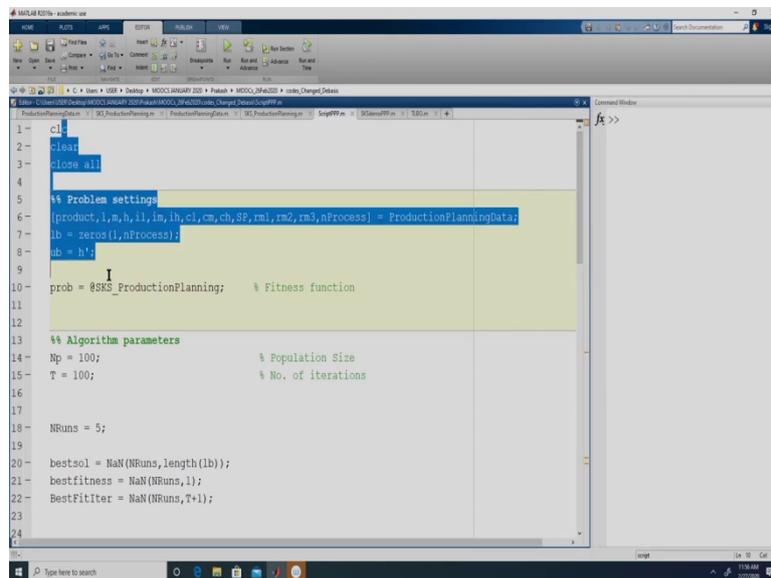


(Refer Slide Time: 59:34)



Let me just remove this breakpoint and let us see the figure directly right. So, this is the convergence curve. As you might remember right so, the sarcastic techniques are to be run multiple times right with multiple runs have to be done by changing the seed of the random number generator right. So, that is what we will do now on this problem.

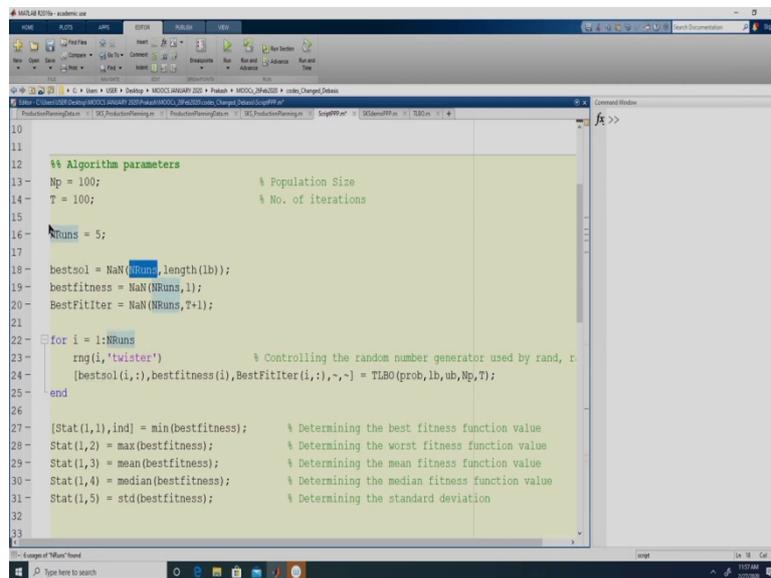
(Refer Slide Time: 59:50)



```
1 - clear
2 - close all
3 -
4 - %% Problem settings
5 - [product, l, m, b, l1, l2, l3, l4, cl, cm, ch, SP, rml, rm2, rm3, nProcess] = ProductionPlanningData;
6 - lb = zeros(1, nProcess);
7 - ub = h';
8 -
9 -
10 - prob = @SKS_ProductionPlanning; % Fitness function
11 -
12 - %% Algorithm parameters
13 -
14 - Np = 100; % Population Size
15 - T = 100; % No. of iterations
16 -
17 -
18 - NRuns = 5;
19 -
20 - bestsol = NaN(NRuns, length(lb));
21 - bestfitness = NaN(NRuns, 1);
22 - BestFitter = NaN(NRuns, T+1);
23 -
24 -
```

So, let us do that right. So, here what we have done is the major part of the code remains the same right. So, this we have discussed right. So, this is these 3, these 4 lines help us to get the problem details right.

(Refer Slide Time: 60:07)

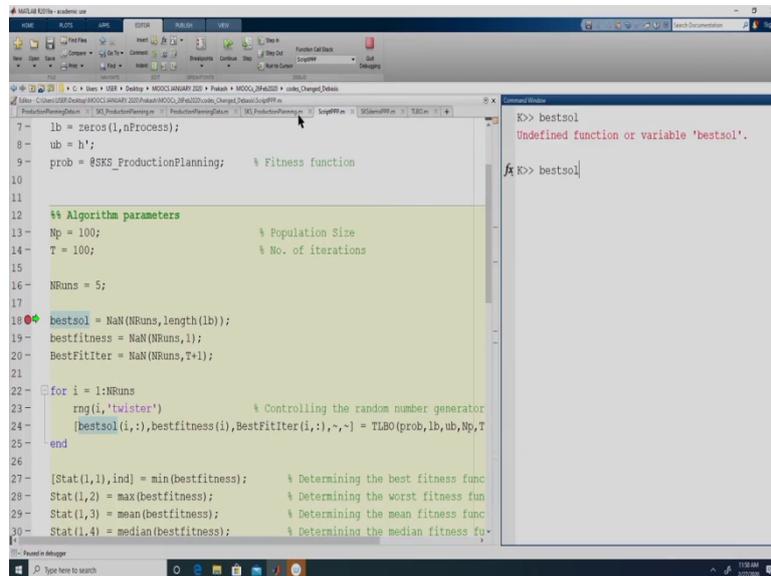


```
10
11
12 %% Algorithm parameters
13 Np = 100;           % Population Size
14 T = 100;           % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFIter = NaN(NRuns, T+1);
21
22 for i = 1:NRuns
23     rng(i, 'twister')           % Controlling the random number generator used by rand, r
24     [bestsol(i,:), bestfitness(i), BestFIter(i,:), -, -] = TLBO(prob, lb, ub, Np, T);
25 end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness function value
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
```

This is the algorithm parameters and let us say now we choose to run 5 times. Just to demonstrate it to you we are restricting our self with 5 runs. This best sol vector previously when we had contains the decision variable right. So, and it is going to contain 54 decision variables even for a single run right. So, what we are doing is here we are defining the matrix bestsol. So, it will have 5 rows because the number of runs is 5 and the number of columns is the length of lb

So, the length of lb is 54 for the current set of data right, but if we change this data this will appropriately change with respect to the number of decision variables right. So, that is this bestsol.

(Refer Slide Time: 60:46)



```
7 - lb = zeros(1,nProcess);
8 - ub = h';
9 - prob = 8SXS_ProductionPlanning; % Fitness function
10
11
12 %% Algorithm parameters
13 - Np = 100; % Population Size
14 - T = 100; % No. of iterations
15
16 - NRuns = 5;
17
18 - bestsol = NaN(NRuns,length(lb));
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23 -     rng(i,'twister') % Controlling the random number generator
24 -     [bestsol(i,:),bestfitness(i),BestFitter(i,:),-] = TLBO(prob,lb,ub,Np,T
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness func
28 - Stat(1,2) = max(bestfitness); % Determining the worst fitness fun
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fitness func
30 - Stat(1,4) = median(bestfitness); % Determining the median fitness fu
```

Command Window

```
K> bestsol
Undefined function or variable 'bestsol'.
fx K> bestsol]
```

So, bestsol if we look at it is the line has not been executed right.

(Refer Slide Time: 60:57)

```
10
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFitIter = NaN(NRuns, T+1);
21
22 for i = 1:NRuns
23     rng(i, 'twister') % Controlling the random number generator
24     [bestsol(i,:), bestfitness(i), BestFitIter(i,:), -, -] = TLBO(prob, lb, ub, Np, T)
25 end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness func
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness func
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness func
30 Stat(1,4) = median(bestfitness); % Determining the median fitness fu
31 Stat(1,5) = std(bestfitness); % Determining the standard deviatio
32
33
```

Command Window

Columns 46 through 48

NaN	NaN	NaN

Columns 49 through 51

NaN	NaN	NaN

Columns 52 through 54

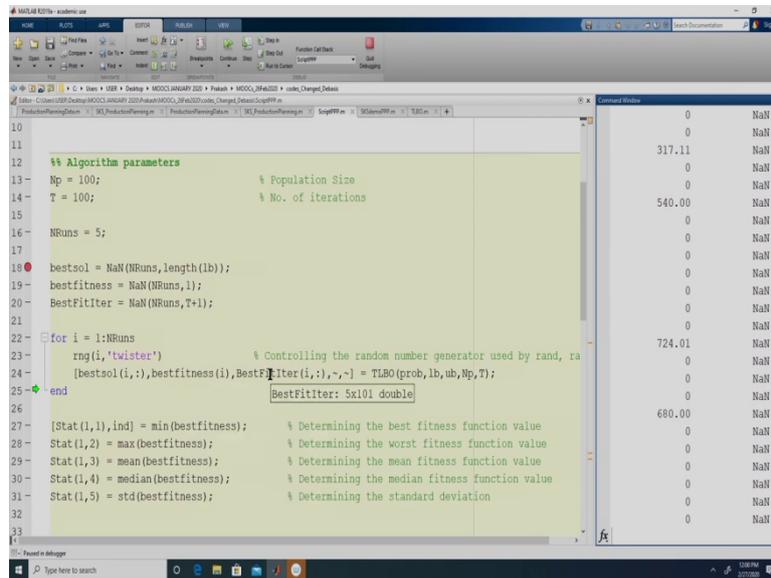
NaN	NaN	NaN
NaN	NaN	NaN
NaN	NaN	NaN

So, f 10 so, this is bestsol. So, it will have 54 columns and 5 rows ok. So, fitness corresponding to each of the solution will be stored in best fitness right. So, this will be a vector of 5 rows and 1 column. So, this convergence curve right will have T plus 1 values right.

So, T is the number of iterations, it will have the best fitness function value for the 100 iteration. In addition it also will have the best fitness function value for the initial population right. So, that is why this is T plus 1 right and we are executing 5 runs right. So, this BestFitIter is a matrix of 5 comma 101 in this case or T plus 1 f 10 right. So, we are executing it for 5 times. Every time we need to change the seed so, we are saying that we will run the twister algorithm with the seed as first as 1, second time as 2, third time as 3, fourth time as 4 and the fifth time as 5.



(Refer Slide Time: 62:11)



The screenshot shows the MATLAB Editor interface. The main window displays a script with the following code:

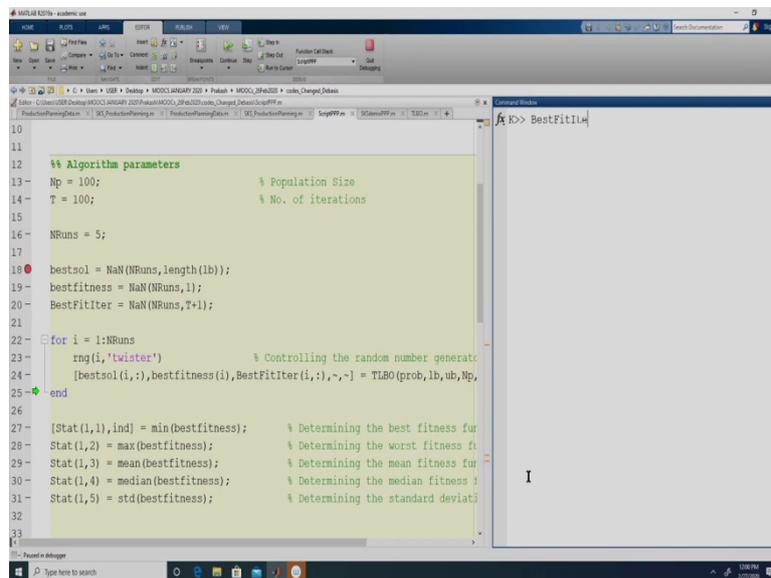
```
10
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFitter = NaN(NRuns, 1);
21
22 for i = 1:NRuns
23     rng(i, 'twister') % Controlling the random number generator used by rand, ra
24     [bestsol(i,:), bestfitness(i), BestFitter(i,:), S, ~] = TLBO(prob, lb, ub, Np, T);
25 end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness function value
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
```

The Command Window on the right shows the following output:

0	NaN
0	NaN
317.11	NaN
0	NaN
540.00	NaN
0	NaN
724.01	NaN
0	NaN
0	NaN
680.00	NaN
0	NaN

So, it has the values in the first column right, best fitness. Again if we see over here, the first value is minus 413.80 which is the same when we ran it for a single run right because that time we had use the same algorithm and seed was fixed 1 right. So, this best fitness is minus 413.80 for the first time. Similarly we will have this convergence curve.

(Refer Slide Time: 62:35)

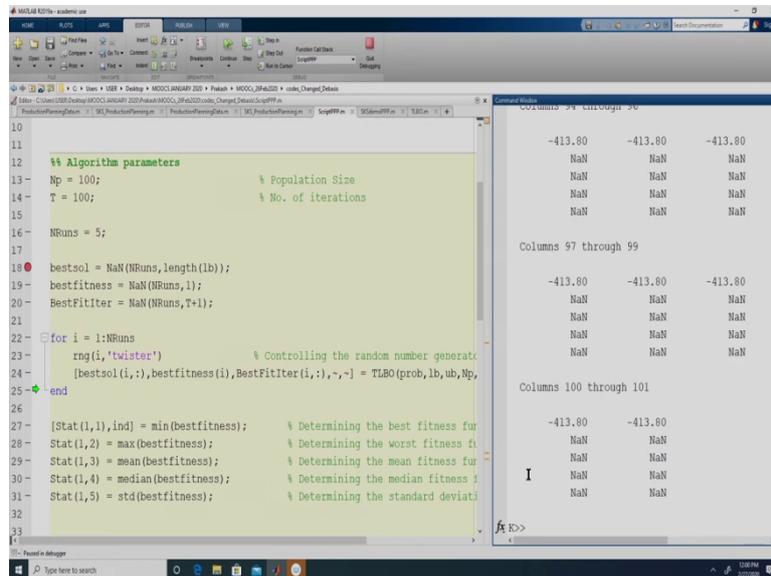


```
10
11
12 %% Algorithm parameters
13 Np = 100;           % Population Size
14 T = 100;           % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFitIter = NaN(NRuns, 1);
21
22 for i = 1:NRuns
23     rng(i, 'twister')           % Controlling the random number generator
24     [bestsol(i,:), bestfitness(i), BestFitIter(i,:), S, -] = TLBO(prob, lb, ub, Np,
25     end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness function value
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation of fitness function
32
33
```

Command Window: K>> BestFitIter

In this case the first row would be populated right. So, BestFitIter right.

(Refer Slide Time: 62:40)



```
10
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFitIter = NaN(NRuns, 1);
21
22 for i = 1:NRuns
23     rng(i, 'twister') % Controlling the random number generator
24     [bestsol(i,:), bestfitness(i), BestFitIter(i,:), S, -] = TLBO(prob, lb, ub, Np,
25     end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness function
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
```

Command Window

-413.80	-413.80	-413.80
NaN	NaN	NaN

Columns 97 through 99

-413.80	-413.80	-413.80
NaN	NaN	NaN

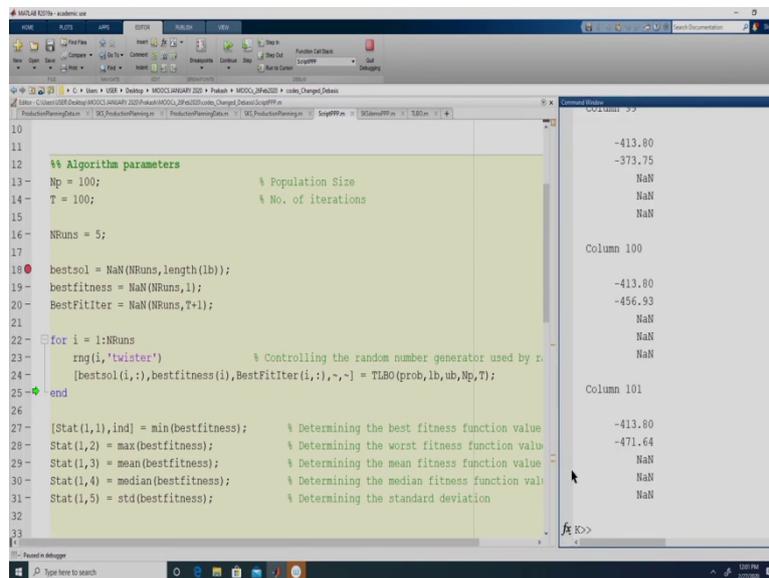
Columns 100 through 101

-413.80	-413.80
NaN	NaN

So, the first row is populated because the first row contains the convergence curve for the first run right. So, in this case, we are not looking at the final population and its fitness function. If you are interested you can save the final population and the fitness function of the final population in each run right. So, this is the second run right. So, this is executing it for the second time.

So, now if we see 2 values are populated. So, first time we got minus 413.8, second time we got minus 470.64 right. Similarly this BestFitIter will now have 2 rows which are populated right.

(Refer Slide Time: 63:16)



The screenshot shows the MATLAB Editor interface. The script in the editor contains the following code:

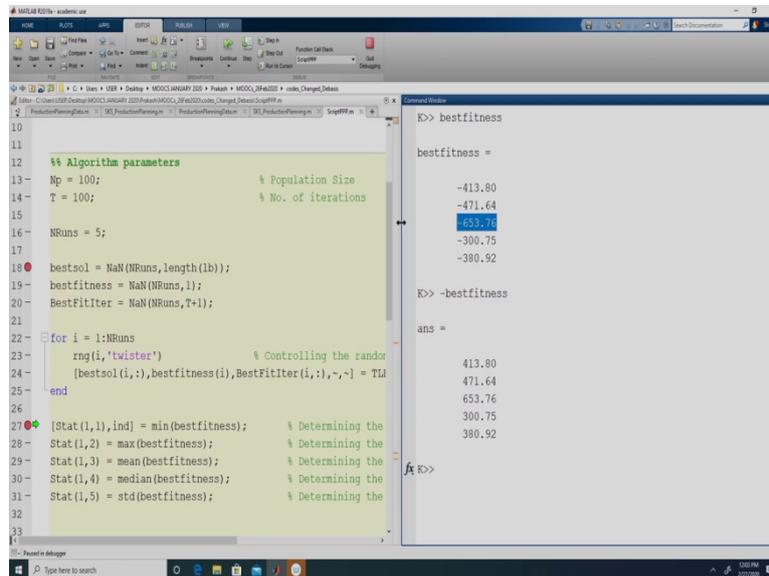
```
10
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFIter = NaN(NRuns, T+1);
21
22 for i = 1:NRuns
23     rng(i, 'twister') % Controlling the random number generator used by r
24     [bestsol(i,:), bestfitness(i), BestFIter(i,:), -, -] = TLBO(prob, lb, ub, Np, T);
25 end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness function value
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
```

The Command Window shows the following output:

```
-413.80
-373.75
NaN
NaN
NaN
NaN
Column 100
-413.80
-456.93
NaN
NaN
NaN
Column 101
-413.80
-471.64
NaN
NaN
NaN
```

So, here if we see 2 rows are populated right. So, similarly the rest of the 3 runs could be executed right. So, we will just put a breakpoint over here and continue ah, it is executing the rest of the 3 runs right.

(Refer Slide Time: 63:35)



```
10
11
12 %% Algorithm parameters
13 Np = 100;           % Population Size
14 T = 100;           % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns, length(lb));
19 bestfitness = NaN(NRuns, 1);
20 BestFitIter = NaN(NRuns, 1);
21
22 for i = 1:NRuns
23     rng(i, 'twister') % Controlling the random
24     [bestsol(i,:), bestfitness(i), BestFitIter(i,:), ~] = TL
25 end
26
27 [Stat(1,1), ind] = min(bestfitness); % Determining the
28 Stat(1,2) = max(bestfitness); % Determining the
29 Stat(1,3) = mean(bestfitness); % Determining the
30 Stat(1,4) = median(bestfitness); % Determining the
31 Stat(1,5) = std(bestfitness); % Determining the
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
K> bestfitness
bestfitness =
-413.80
-471.64
653.76
-300.75
-380.92
K> -bestfitness
ans =
413.80
471.64
653.76
300.75
380.92
K>
```

So, here let us wait for the command prompt. So, we have a command prompt now over here. So, best fitness if we see it has 5 values right. So, this is the best solution which we got in the first run, this is the best solution which we got in the second run, this is a fitness function value of the best solution in the third run and the fourth run and the fifth run right.

So, right now if we see third run gives us the best solution right because it has the lowest value right. So, this best fitness is the fitness function value and it is for minimization right. If you are talking about profit then these are the profit right. So, the maximum profit we get is 653.6 right or the minimum fitness is minus 653.76.

So, the third run gives us the best solution. We want to extract that particular solution right ultimately it is the solution which will help us to realize the objective function value right. So, if someone tells us that profit is 653.76 for some production plan. It is not useful to us right

because unless we know the values of the decision variables, the value of the objective function hardly helps us to implement it right. So, it is necessary for us to get the decision variables right and the decision variables are in this bestsol right.

(Refer Slide Time: 64:50)

```

10
11
12 %% Algorithm parameters
13 Np = 100; % Population Size
14 T = 100; % No. of iterations
15
16 NRuns = 5;
17
18 bestsol = NaN(NRuns,length(lb));
19 bestfitness = NaN(NRuns,1);
20 BestFitter = NaN(NRuns,T+1);
21
22 for i = 1:NRuns
23     rng(i,'twister') % Controlling the random number generator
24     [bestsol(i,:),bestfitness(i),BestFitter(i,:),-,] = TLBO(prob,lb,ub,
25     end
26
27 [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness
30 Stat(1,4) = median(bestfitness); % Determining the median fitness
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33

```

```

Command Window
471.64
653.76
300.75
380.92

E>> bestsol(3,:)

ans =

Columns 1 through 3
    16.00    0    0

Columns 4 through 6
    0    0    0

Columns 7 through 9
    0    0    0

Columns 10 through 12
    0    0    0

```

So, the best sol we are looking at the third row all the columns that is what we are interested right. So, third row all the columns is given over here. So, for the third run the best solution is process 1 is to be used to produce 270, process 17 right; this is 16, this is 17, this is 18 because it write 16 to 18 over here.

(Refer Slide Time: 65:01)

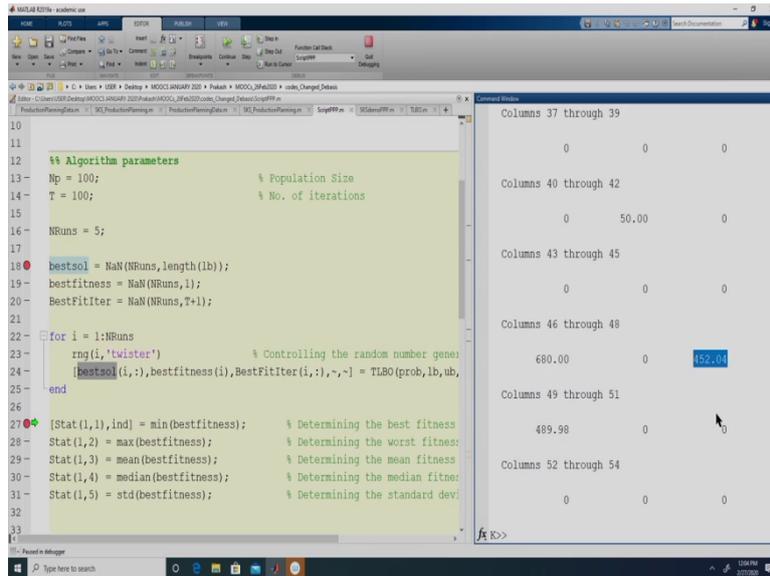
The image shows a MATLAB Editor window with a script and its Command Window output. The script defines algorithm parameters and performs a genetic algorithm. The Command Window shows the output of the script, which is a 1x5 matrix of statistics.

```
10  
11  
12 %% Algorithm parameters  
13 Np = 100; % Population Size  
14 T = 100; % No. of iterations  
15  
16 NRuns = 5;  
17  
18 bestsol = NaN(NRuns, length(lb));  
19 bestfitness = NaN(NRuns, 1);  
20 BestFIter = NaN(NRuns, 1);  
21  
22 for i = 1:NRuns  
23     rng(i, 'twister') % Controlling the random number gene;  
24     [bestsol(i,:), bestfitness(i), BestFIter(i,:), -, -] = TLBO(prob, lb, ub,  
25     end  
26  
27 [Stat(1,1), ind] = min(bestfitness); % Determining the best fitness  
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness  
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness  
30 Stat(1,4) = median(bestfitness); % Determining the median fitness  
31 Stat(1,5) = std(bestfitness); % Determining the standard dev;  
32  
33
```

Command Window Output:

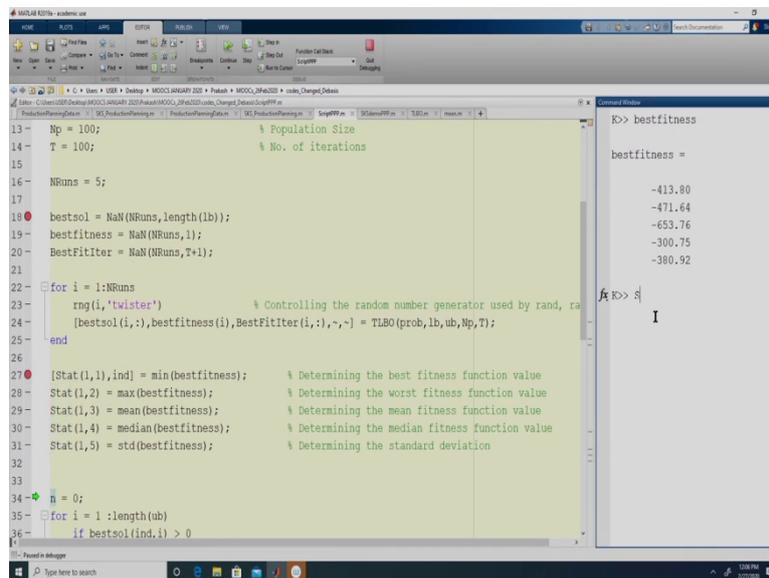
Columns	Value
Columns 4 through 6	0 0 0
Columns 7 through 9	0 0 0
Columns 10 through 12	0 0 0
Columns 13 through 15	0 0 0
Columns 16 through 18	0 200.00 0
Columns 19 through 21	0 0 0

(Refer Slide Time: 65:11)



So, process 17 is to be used to produce 200 and then process 41 is to be used to produce 50 process 46 and 48 are used to produce 68 and 452.04 and process 49 is to be used to produce 489.98 right. So, that is the production plan.

(Refer Slide Time: 65:31)



```
13 - Np = 100; % Population Size
14 - T = 100; % No. of iterations
15
16 - NRuns = 5;
17
18 - bestsol = NaN(NRuns,length(lb));
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23 -     rng(1,'twister') % Controlling the random number generator used by rand, ra
24 -     [bestsol(i,:),bestfitness(i),BestFitter(i,:),:-] = TLBO(prob,lb,ub,Np,T);
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function value
28 - Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 - Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 - Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 - n = 0;
35 - for i = 1 :length(ub)
36 -     if bestsol(ind,i) > 0
```

```
K>> bestfitness
bestfitness =
-413.80
-471.64
-653.76
-300.75
-380.92
```

So, we can directly determine that by having this small piece of code right. Remember for this best fitness again we are supposed to do a statistical analysis. So, here in line 27 what we are saying is we are finding out the best fitness value right in best fitness. So, out of the 5 runs which is the runs that gives us the best solution since we have converted the problem into minimization right. So, min of best fitness gives us the best solution right. So, that value would be stored in Stat of 1. Remember this value Stat which we had previously used for doing statistical analysis right.

So, it is going to be a row vector, the first column is going to have the best run, the second column is going to have the worst run, the third column is going to give us mean of all the runs, the fourth column is going to give us the median and the fifth column is going to give us the standard deviation right.

So, that is how we had arranged it previously. Stat will be a row vector; first column is best solution, second column worst solution, third column mean, fourth column median and fifth column is the standard deviation in this 5 runs right. So, here what we are doing in addition to getting this value right of minus 653.76, we also getting the location which run it is occurring right. So, that will be stored in ind.

Remember this mean function we have used multiple times in addition to giving the best value, it can also give us where it is located right. So, where it is located is required so, that we can extract that solution from this bestsol right. So, that is why we are having this ind over here, it is just name of a variable.

(Refer Slide Time: 67:08)

The screenshot shows a MATLAB script in the Editor and its execution output in the Command Window. The script defines parameters for a fitness function optimization process, including population size (Np), iterations (T), and number of runs (NRuns). It then performs 5 runs, storing the results in a matrix 'bestsol'. The script calculates various statistics from 'bestsol': the best fitness value and its location (ind), the worst fitness value, the mean, median, and standard deviation of the fitness values across the 5 runs.

```

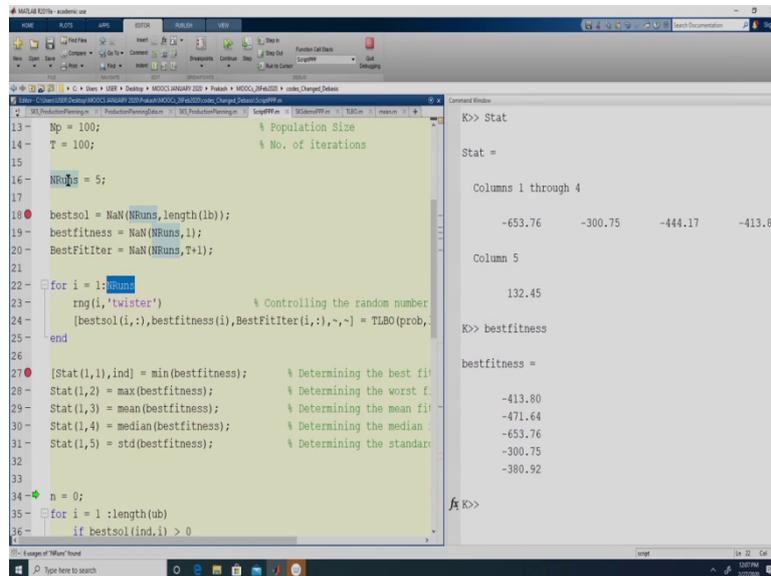
13 - Np = 100; % Population Size
14 - T = 100; % No. of iterations
15
16 - NRuns = 5;
17
18 - bestsol = NaN(NRuns,length(lb));
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23 -     rng(i,'twister') % Controlling the random number generator used by rand, ra
24 -     [bestsol(i,:),bestfitness(i),BestFitter(i,:),:-] = TLBO(prob,lb,ub,Np,T);
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function value
28 - Stat(1,2) = max(bestfitness); % Determining the worst fitness function value
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fitness function value
30 - Stat(1,4) = median(bestfitness); % Determining the median fitness function value
31 - Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 - h = 0;
35 - for i = 1 :length(ub)
36 -     if bestsol(ind,i) > 0

```

The Command Window output shows the 'bestfitness' values for 5 runs: -413.80, -471.64, -653.76, -300.75, and -380.92. It also shows the 'Stat' variable as a row vector: [-653.76, -300.75, -444.17, -413.80, 132.45].

So, if we just execute this so, now if we see this variable Stat.

(Refer Slide Time: 67:10)



The screenshot shows the MATLAB Editor interface. The script on the left contains the following code:

```
13 - Np = 100; % Population Size
14 - T = 100; % No. of iterations
15
16 - NRuns = 5;
17
18 - bestsol = NaN(NRuns,length(lb));
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23 -     rng(1,'twister') % Controlling the random number
24 -     [bestsol(i,:),bestfitness(i),BestFitter(i,:),-]= TLBO(prob,
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fit
28 - Stat(1,2) = max(bestfitness); % Determining the worst fit
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fitness
30 - Stat(1,4) = median(bestfitness); % Determining the median fitness
31 - Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 - n = 0;
35 - for i = 1:length(ub)
36 -     if bestsol(ind,i) > 0
```

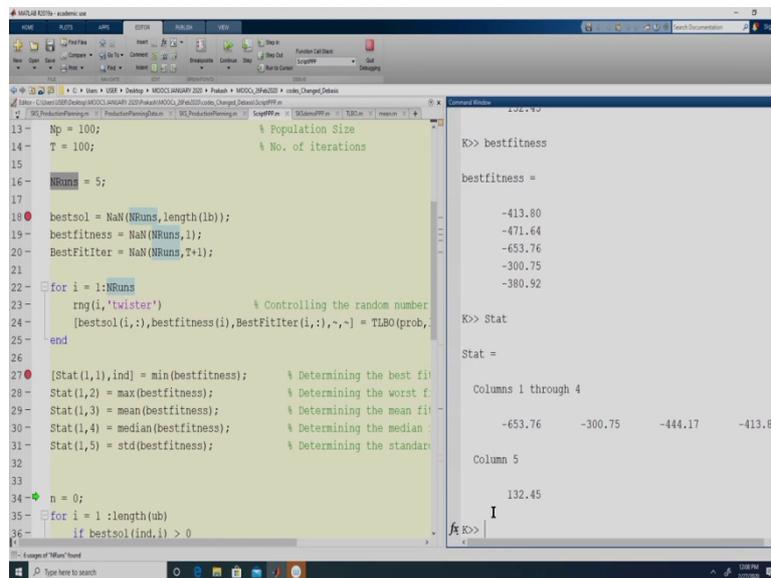
The Command Window on the right shows the following output:

```
K>> Stat
Stat =
Columns 1 through 4
-653.76 -300.75 -444.17 -413.80
Column 5
132.45
K>> bestfitness
bestfitness =
-413.80
-471.64
-653.76
-300.75
-380.92
```

So, the best run is minus 653.76, the worst is minus 300.75 the mean of the 5 runs is minus 444.17, the median of the 4 runs is minus 413.8 and the standard deviation is 132.45.

So, here you can see the standard deviation can be significantly higher right or even best fitness the difference is quite significant right. We can get as low as minus 300 and as high as minus 653.76. So, if we increase the number of runs to let us say from 5 to let us say 25, we may even get a better solution than minus 653.76 right. So, that is why it is important to run multiple times.

(Refer Slide Time: 67:56)



```
13 - Np = 100; % Population Size
14 - T = 100; % No. of iterations
15
16 - NRuns = 5;
17
18 - bestsol = NaN(NRuns,length(lb));
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23 -     rng(1,'twister') % Controlling the random number
24 -     [bestsol(i,:),bestfitness(i),BestFitter(i,:),-,-] = TLBO(prob,
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fit
28 - Stat(1,2) = max(bestfitness); % Determining the worst fit
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fit
30 - Stat(1,4) = median(bestfitness); % Determining the median
31 - Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 - n = 0;
35 - for i = 1:length(ub)
36 -     if bestsol(ind,i) > 0
```

```
K> bestfitness
bestfitness =
-413.80
-471.64
-653.76
-300.75
-380.92

K> Stat
Stat =
Columns 1 through 4
-653.76 -300.75 -444.17 -413.80
Column 5
132.45
```

So, this is the stat variable, it gives the statistical analysis of all the 5 runs right. So, here we have plugged it with TLBO, you can use any of the other 4 meta heuristic techniques that we have discussed. Nothing would change right, you need to just call the appropriate function and give the appropriate input to the materialistic techniques right.

So, these 3 input is going to remain constant because that is coming from the optimization problem which we are solving. So, these 2 would also remain the same for all the 4 meta heuristic techniques which we have discussed right. But if you use simulated annealing or something else you might have to give a different value over here otherwise the rest of the parameters, you need to appropriately give.

So, for example, if you are choosing differential evolution, then you will also have to give the crossover probability and the scaling factor. If you are choosing to solve it with particle swarm

optimization in addition to  $N$   $p$  and  $T$  you will have to give the inertia weight and the acceleration coefficient  $c_1$  and  $c_2$  right. So, the rest of the discussion is going to just involve on post optimality analysis right.

(Refer Slide Time: 69:00)

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

```

19- bestfitness = NaN(NRuns,1);
20- Bestfitter = NaN(NRuns,T+1);
21-
22- for i = 1:NRuns
23-     rng(i,'twister') % Controlling the random number generator used by
24-     [bestsol(i,:),bestfitness(i),Bestfitter(i,:),-,] = TLBO(prob,lb,ub,lb,ub);
25- end
26-
27- [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function val
28- Stat(1,2) = max(bestfitness); % Determining the worst fitness function va
29- Stat(1,3) = mean(bestfitness); % Determining the mean fitness function val
30- Stat(1,4) = median(bestfitness); % Determining the median fitness function v
31- Stat(1,5) = std(bestfitness); % Determining the standard deviation
32-
33-
34- n = 0;
35- for i = 1:length(ub)
36-     if bestsol(ind,i) > 0
37-         n = n + 1;
38-         disp(sol(n,:)) = [product(i) bestsol(ind,i)];
39-     end
40- end
41-
42- % bestsol = bestsol(ind,:);

```

The Command Window shows the following output:

```

Columns 49 through 50
    0         0
    0         0
  489.98      0
    0         0
  680.00     23.38

Columns 51 through 52
    0         0
  30.74      0
    0         0
    0         0
    0         0

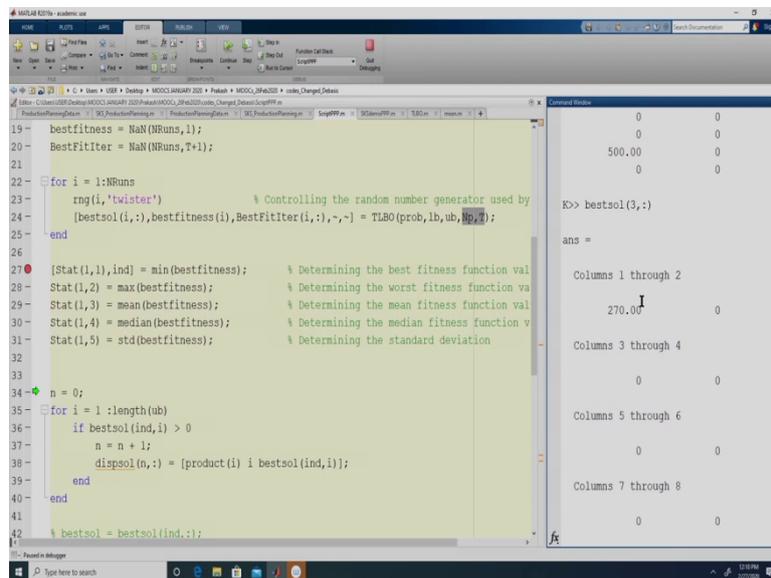
Columns 53 through 54
    0         0
    0         0
    0         0
  500.00      0
    0         0

```

The Command Window prompt shows: `R>> bestsol(3,)`

So, what we are doing over here is this best solution if we see it contains; so, the third run is what we are interested right.

(Refer Slide Time: 69:09)



```
19 - bestfitness = NaN(NRuns,1);
20 - BestFitter = NaN(NRuns,T+1);
21
22 - for i = 1:NRuns
23     rng(i,'twister') % Controlling the random number generator used by
24     [bestsol(i,:),bestfitness(i),BestFitter(i,:),-,-] = TLBO(prob,lb,ub,Np,T);
25 - end
26
27 - [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function val
28 - Stat(1,2) = max(bestfitness); % Determining the worst fitness function va
29 - Stat(1,3) = mean(bestfitness); % Determining the mean fitness function val
30 - Stat(1,4) = median(bestfitness); % Determining the median fitness function v
31 - Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 - n = 0;
35 - for i = 1 :length(ub)
36     if bestsol(ind,i) > 0
37         n = n + 1;
38         dispSol(n,:) = [product(i) i bestsol(ind,i)];
39     end
40 - end
41
42 - % bestsol = bestsol(ind,:);
```

Command Window

```
0 0
0 0
500.00 0
0 0

K> bestsol(3,:)

ans =

Columns 1 through 2

270.00 0

Columns 3 through 4

0 0

Columns 5 through 6

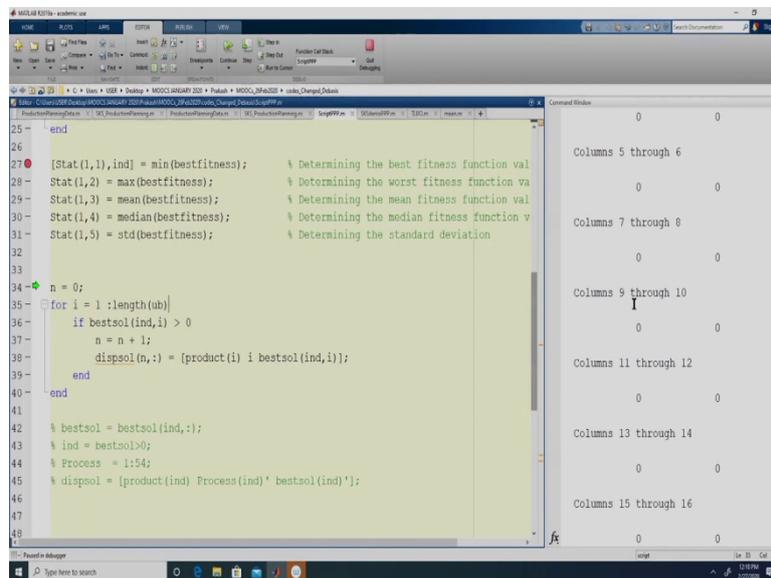
0 0

Columns 7 through 8

0 0
```

So, right now it is displaying all the 0, let us say we want to display only the non zero values right along with the product as well as the process number right.

(Refer Slide Time: 69:21)



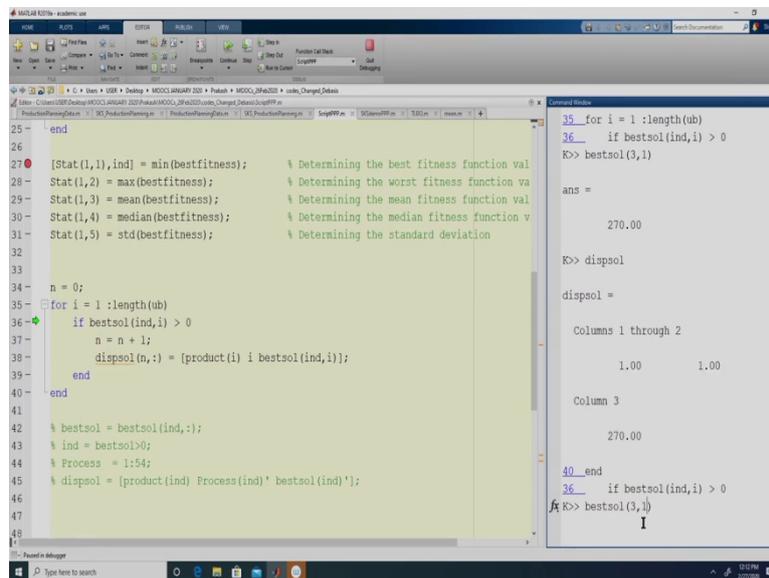
```
25 - end
26
27 [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function val
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function va
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function val
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function v
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 n = 0;
35 for i = 1 :length(ub)
36     if bestsol(ind,i) > 0
37         n = n + 1;
38         disp(sol(n,:)) = [product(i) i bestsol(ind,i)];
39     end
40 end
41
42 % bestsol = bestsol(ind,:);
43 % ind = bestsol>0;
44 % Process = 1:54;
45 % disp(sol = [product(ind) Process(ind)* bestsol(ind)'];
46
47
48
49
```

Command Window

0	0
Columns 5 through 6	
0	0
Columns 7 through 8	
0	0
Columns 9 through 10	
0	0
Columns 11 through 12	
0	0
Columns 13 through 14	
0	0
Columns 15 through 16	
0	0

So, then we can write a small piece of code right which is what we have done over here right.

(Refer Slide Time: 69:24)



```
25 - end
26
27 [Stat(1,1),ind] = min(bestfitness); % Determining the best fitness function val
28 Stat(1,2) = max(bestfitness); % Determining the worst fitness function va
29 Stat(1,3) = mean(bestfitness); % Determining the mean fitness function val
30 Stat(1,4) = median(bestfitness); % Determining the median fitness function v
31 Stat(1,5) = std(bestfitness); % Determining the standard deviation
32
33
34 n = 0;
35 for i = 1 :length(ub)
36     if bestsol(ind,i) > 0
37         n = n + 1;
38         dispsol(n,:) = [product(i) i bestsol(ind,i)];
39     end
40 end
41
42 % bestsol = bestsol(ind,:);
43 % ind = bestsol(0);
44 % Process = 1:54;
45 % dispsol = [product(ind) Process(ind)* bestsol(ind)];
46
47
48
49
```

```
35 for i = 1 :length(ub)
36     if bestsol(ind,i) > 0
K> bestsol(3,1)
ans =
    270.00
K> dispsol
dispsol =
Columns 1 through 2
    1.00    1.00
Column 3
    270.00
40 end
36     if bestsol(ind,i) > 0
K> bestsol(3,1)
    I
```

So, what we are doing is we are initializing scalar n is equal to 0 right. So, this scalar is going to keep track of the solutions which are non zero right. So, and we run a loop across the decision variables right.

So, we have 54 decision variables. So, this length of this lb would be 54. So, we are just going one variable by one variable right and we know that the best solution is in this row, ind right. So, in this variable bestsol, it will contain 5 cross 4 we are interested in the third row right and we are checking for the i th variable. So, if it is greater than 0, we are increasing the counter of n by 1 and we are creating this dispsol right a matrix dispsol.

So, this is just to display the solution right. So, what we are doing here is we are stacking the product number right. This variable product contains which process produces which product right. So, we are saying product of i right what is the ith value in this vector product and the

value  $i$  right the value  $i$  because the location  $i$  indicates the process number that is how we have defined our decision variable and this `bestsol` of `ind` comma  $i$  will give us the value the exact value right.

So, for example, now  $i$  is 1 right. So, if we say what is best sol of 3 comma one right. So, `bestsol` of 3 comma 1 is 270 right. So, it will put 270 over here right and it will come inside this loop only if this value is non zero right. If it had been a 0 value, it would not come inside this if condition, it will not satisfy this if condition and it will not come over here right. So, this we run across all the decision variable right. So, `dispsol` if we see now, it says that the first product is produced from process 1 and the quantity produced is 270 right.

So, this has to be run for all the processes. So, second time this value is actually 0 right. So, `bestsol` 3 of 2 would be 0 right.

(Refer Slide Time: 71:26)

The screenshot shows the MATLAB Editor interface. The main window displays a script with the following code:

```

38- disp(n,:) = [product(i) bestsol(ind,i)];
39- end
40- end
41
42 % bestsol = bestsol(ind,:);
43 % ind = bestsol>0;
44 % Process = 1:54;
45 % dispsol = [product(ind) Process(ind) bestsol(ind)];
46
47
48
49 for i = 1:NRuns
50     % ind = BestFitter(i,:)>0;
51     % BestFitter(i,ind) = NaN;
52     % plot(1:T+1,BestFitter(i,:), 'r')
53     n = 0;
54     for j = 1:T+1
55         if BestFitter(i,j) < 0
56             n = n + 1;
57             y(n) = BestFitter(i,j);
58             x(n) = j;
59         end
60     end
61

```

The Command Window on the right shows the following output:

```

270.00
K> disp(n)
dispsol =
Columns 1 through 2
    1.00    1.00
Column 3
    270.00
40_end
36_ if bestsol(ind,i) > 0
K> bestsol(3,2)
ans =
    0
40_end
K>

```

So, it will not get into this if condition. So, n still remains one though we have checked 2 variable only the first variable went through this if condition because the production was greater than 0, this loop gets executed for 54 times right. So, let me just put over here and then execute it right.

(Refer Slide Time: 71:47)

```

38 dispPP = 0;
39 disp = [0 0 0];
40 end
41
42 % bestsol = be
43 % ind = bestsol
44 % Process = 1:
45 % dispsol = [p
46
47
48
49 for i = 1:NRun
50     % ind
51     % Best
52     % plot
53     n = 0;
54     for j = 1:3
55         if Best
56             n =
57             y(f
58             x(f
59         end
60     end

```

Column 3

21.00	46.00	
21.00	48.00	
21.00	49.00	
270.00		
200.00		
50.00		
680.00		
452.04		
489.98		

K> dispsol

dispsol =

1.00	1.00	270.00
7.00	17.00	200.00
19.00	41.00	50.00
21.00	46.00	680.00
21.00	48.00	452.04
21.00	49.00	489.98

So, now we can just have a look at this dispsol right. So, instead of trying to figure out which are the processes producing manually, we have just made use of couple of lines of code to come up with this non zero values.

So, it tells us that the first product is produced from process one and the amount of production is 270 right we produce product 7 from process 17 and the quantity of production is 200. We produced product 19 from process 41 and the production quantity is 50 and

product 21 is produced from process 46, 48 and 49. The production from process 46 is 680, from process 48 is 452 and from 49 is 489.98 right.

(Refer Slide Time: 72:37)

The screenshot shows a MATLAB script in the Editor and its execution results in the Command Window. The script includes a loop for finding the best solution and a final loop for plotting convergence curves. The Command Window displays the output of the 'dispsol' variable, which is a 5x3 matrix of production values.

```

36 ~ if bestsol(ind,i) > 0
37 ~     n = n + 1;
38 ~     dispsol(n,:) = [product(i) i bestsol(ind,i)];
39 ~ end
40 ~ end
41
42 ~ bestsol = bestsol(ind,2);
43 ~ ind = bestsol(3);
44 ~ Process = 1:54;
45 ~ dispsol = [product(ind) Process(ind) * bestsol(ind)];
46 ~
47 ~
48 ~
49 ~ for i = 1:NRuns
50 ~     % ind = BestFitter(i,:)>0;
51 ~     % BestFitter(i,ind) = NaN;
52 ~     % plot(1:T+1,BestFitter(i,:),'r')
53 ~     n = 0;
54 ~     for j = 1:T+1
55 ~         if BestFitter(i,j) < 0
56 ~             n = n + 1;
57 ~             y(n) = BestFitter(i,j);
58 ~             x(n) = j;
59 ~         end

```

Command Window Output:

```

21.00    46.00
21.00    48.00
21.00    49.00

Column 3
270.00
200.00
50.00
680.00
452.04
489.98

K>> dispsol

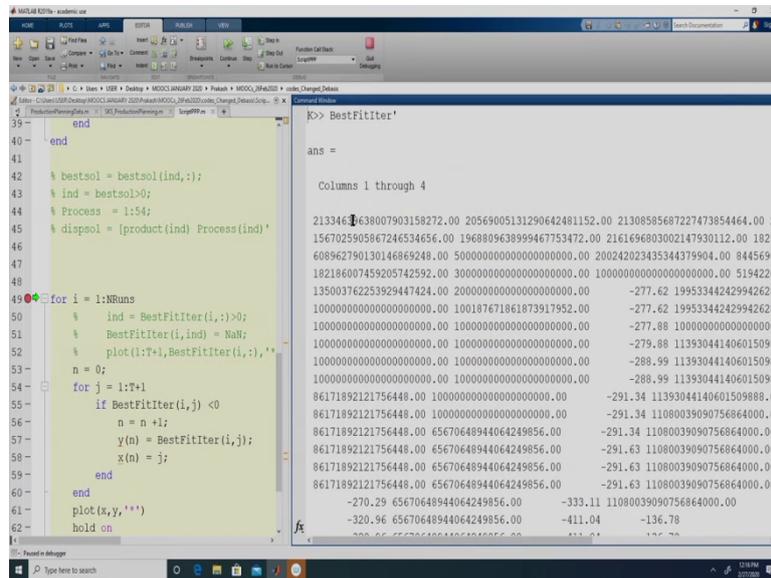
dispsol =
    1.00    1.00   270.00
    7.00   17.00   200.00
   19.00   41.00    50.00
   21.00   46.00   680.00
   21.00   48.00   452.04
   21.00   49.00   489.98

```

So, now we do not have to manually see what are the non zero values right. So, this piece of code can help us to actually get the non zero values right. So, we have explained you this you can also directly use conditional indexing right and also get the same values right. Now that we have analyzed this best solution now let us try to get all the convergence curve on the same plot right. So, since we had solve this problem 5 times right, we can also plot all the 5 convergence curve and now we are not interested to plot the entire convergence curve right; we are only interested to plot the convergence curve from the point it discovers a feasible solution right.



(Refer Slide Time: 73:20)

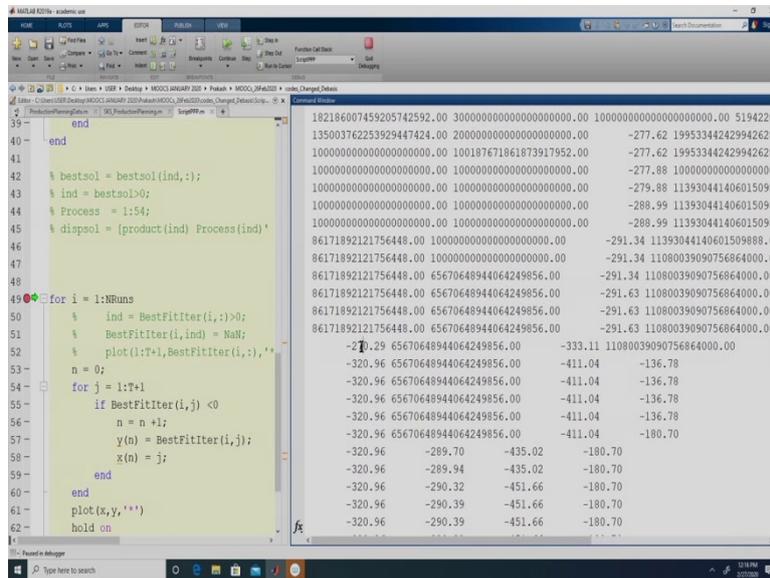


The image shows a MATLAB Editor window with a script and a Command Window showing the output of the script. The script is a function that iterates over a range of values and calls the `BestFitter` function. The Command Window shows the output of the `BestFitter` function, which is a matrix of values.

```
40 - end
41
42 % bestsol = bestsol(ind,:);
43 % ind = bestsol(0);
44 % Process = 1:54;
45 % dispcol = [product(ind) Process(ind)]
46
47
48
49 for i = 1:NRuns
50     % ind = BestFitter(i,:)>0;
51     % BestFitter(i,ind) = NaN;
52     % plot(1:7+i,BestFitter(i,:),**
53     n = 0;
54     for j = 1:7+1
55         if BestFitter(i,j) <0
56             n = n+1;
57             y(n) = BestFitter(i,j);
58             x(n) = j;
59         end
60     end
61     plot(x,y,**)
62     hold on
```

```
K> BestFitter'
ans =
Columns 1 through 4
213346.1638007903158272.00 20569005131290642481152.00 21308595687227473854464.00 2
1567025905867246534656.00 1968809638999467753472.00 2161696803002147930112.00 1825
608962790130146869248.00 50000000000000000000.00 200242023435344379904.00 8445690
182186007459205742592.00 30000000000000000000.00 10000000000000000000.00 5194220
135003762253929447424.00 20000000000000000000.00 -277.62 199533442429942628
10000000000000000000.00 10018761861873917952.00 -277.62 199533442429942628
10000000000000000000.00 10000000000000000000.00 -277.88 10000000000000000000
10000000000000000000.00 10000000000000000000.00 -279.88 11393041406015098
10000000000000000000.00 10000000000000000000.00 -288.99 11393041406015098
10000000000000000000.00 10000000000000000000.00 -288.99 11393041406015098
86171892121756448.00 10000000000000000000.00 -291.34 113930414060150988.0
86171892121756448.00 10000000000000000000.00 -291.34 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.34 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
-270.29 65670648944064249856.00 -333.11 11080039090756864000.0
-320.96 65670648944064249856.00 -411.04 -136.78
```

(Refer Slide Time: 73:23)



The image shows a MATLAB Editor window with a script and a Command Window displaying the output of the script. The script is a loop that iterates over a range of values, performing calculations and plotting. The Command Window shows the results of these calculations, including values for 'ind', 'Process', and 'dispol'.

```
40 end
41
42 % bestsol = bestsol(ind,:);
43 % ind = bestsol(0);
44 % Process = 1:54;
45 % dispol = [product(ind) Process(ind)]
46
47
48
49 for i = 1:NRuns
50     % ind = BestFitter(i,:)>0;
51     % BestFitter(i,ind) = NaN;
52     % plot(1:i, BestFitter(i,:), '**')
53     n = 0;
54     for j = 1:T+1
55         if BestFitter(i,j) < 0
56             n = n + 1;
57             y(n) = BestFitter(i,j);
58             x(n) = j;
59         end
60     end
61     plot(x,y, '**')
62     hold on
```

Command Window Output:

```
182186007459205742592.00 30000000000000000000.00 10000000000000000000.00 5194220
135003762253929447424.00 20000000000000000000.00 -277.62 199533442429942628
10000000000000000000.00 10018761861873917952.00 -277.62 199533442429942628
10000000000000000000.00 10000000000000000000.00 -277.88 10000000000000000000
10000000000000000000.00 10000000000000000000.00 -279.88 113930441406015098
10000000000000000000.00 10000000000000000000.00 -289.99 113930441406015098
10000000000000000000.00 10000000000000000000.00 -289.99 113930441406015098
86171892121756448.00 10000000000000000000.00 -291.34 11393044140601509888.0
86171892121756448.00 10000000000000000000.00 -291.34 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.34 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
86171892121756448.00 65670648944064249856.00 -291.63 11080039090756864000.0
-21.29 65670648944064249856.00 -333.11 11080039090756864000.0
-320.96 65670648944064249856.00 -411.04 -136.78
-320.96 65670648944064249856.00 -411.04 -136.78
-320.96 65670648944064249856.00 -411.04 -136.78
-320.96 65670648944064249856.00 -411.04 -136.78
-320.96 65670648944064249856.00 -411.04 -136.78
-320.96 -289.70 -435.02 -180.70
-320.96 -289.94 -435.02 -180.70
-320.96 -290.32 -451.66 -180.70
-320.96 -290.39 -451.66 -180.70
-320.96 -290.39 -451.66 -180.70
```



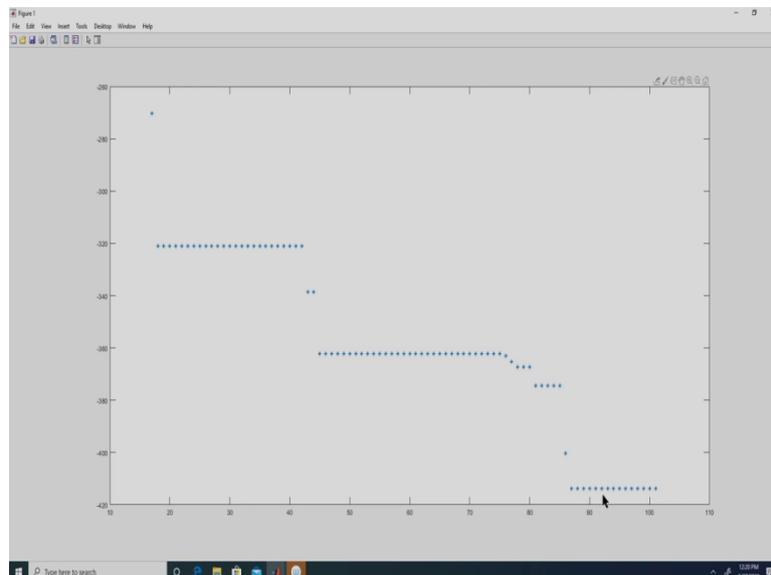


right, similarly second time it did not enter, third time it did not enter, fourth time it did not, fifth time it did not right, sixth time it did not right. So, j value currently is 7.

Similarly j is equal to 8 it did not, 9 it did not 10, it did not; let us just keep moving right. So, j is equal to 17. So, when j is equal 17, we get this value to be less than 0 right. So, we increase the counter of n by 1 right and say y of n is equal to whatever value we have in this case it is minus 270.29. So, here if we see minus 270.29 was the first solution right and n is currently 1 right and this occurs at 17 th value right. So, that is why j is also 17 right.

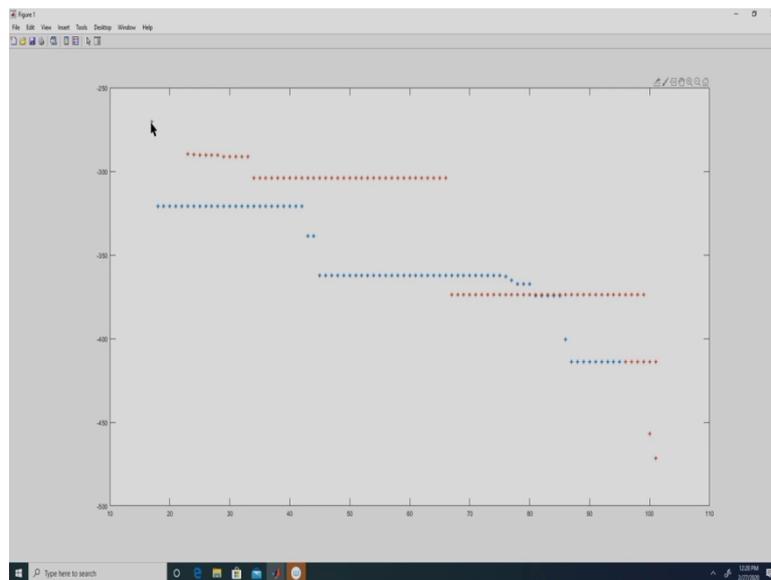
So, in this case we keep track of what is the value and we also keep track of the x. So, this will run for all the iterations for all the iterations T and also the initial population right. So, let me just put a breakpoint over here and then continue right.

(Refer Slide Time: 76:05)



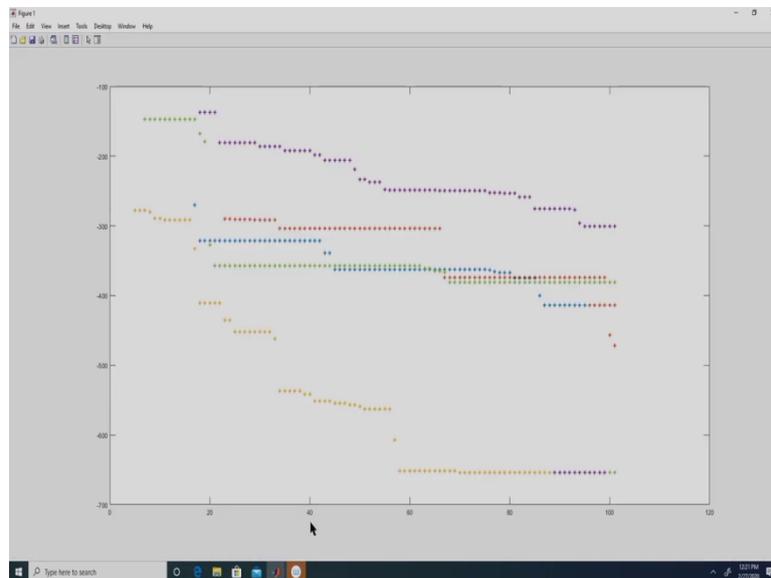
So, this is the convergence curve for the first run. So, x axis is the number of iteration right, y axis is the best fitness function value obtained in that iteration. So, here now we can look at this and analyze whether the algorithm has really converged or additional number of iterations are required.

(Refer Slide Time: 76:30)



So, this if we continue right so, every time it will plot the convergence curve. So, the for the first time, we got a feasible solution in 17, for the second run we got a feasible solution in 23 and the value of minus 289.7 right.

(Refer Slide Time: 76:48)



So, let us just come over here. So, now, we have all the convergence curve. So, that completes this session. In this session what we have done is we have implemented the fitness function file for the production planning problem and then we independently checked it before deploying it with any optimization algorithm.

When we were reasonably sure that the fitness function file is doing things correctly, we plugged it with the optimization algorithm right and then we executed multiple runs on it and then did a statistical analysis right. So, after statistical analysis we just did some post optimality analysis wherein we looked at the convergence curve and we also saw how to plot the a convergence curve. With that will conclude this session.

Thank you.

