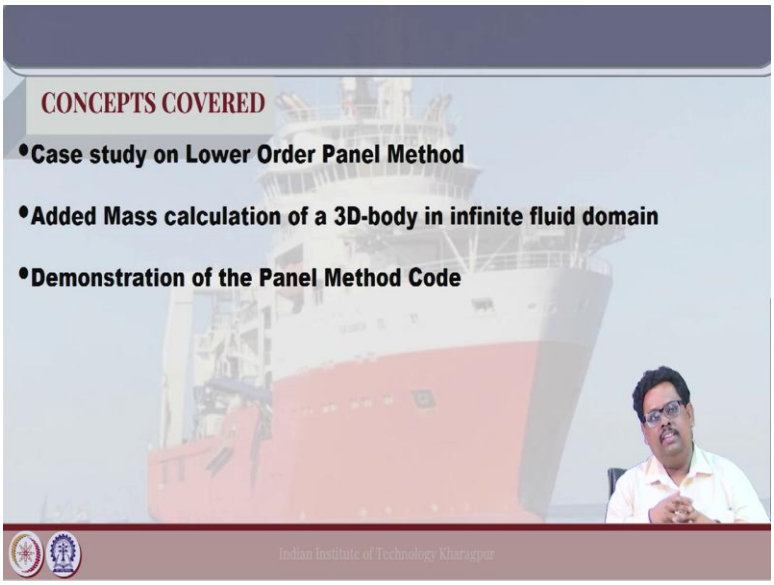
Numerical Ship and Offshore Hydrodynamics
Prof. Ranadev Datta
Department of Ocean Engineering and Naval Architecture
Indian Institute of Technology, Kharagpur

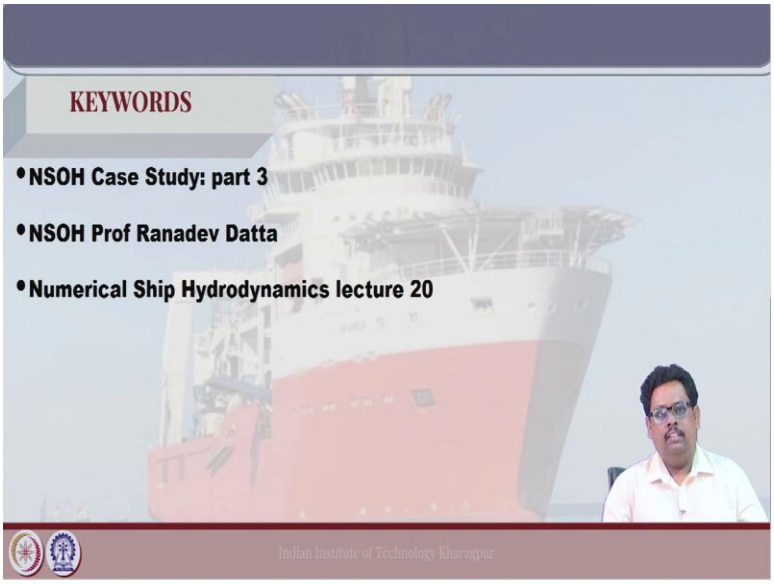
Lecture - 20
Demonstration of Panel Method Code

(Refer Slide Time: 00:20)

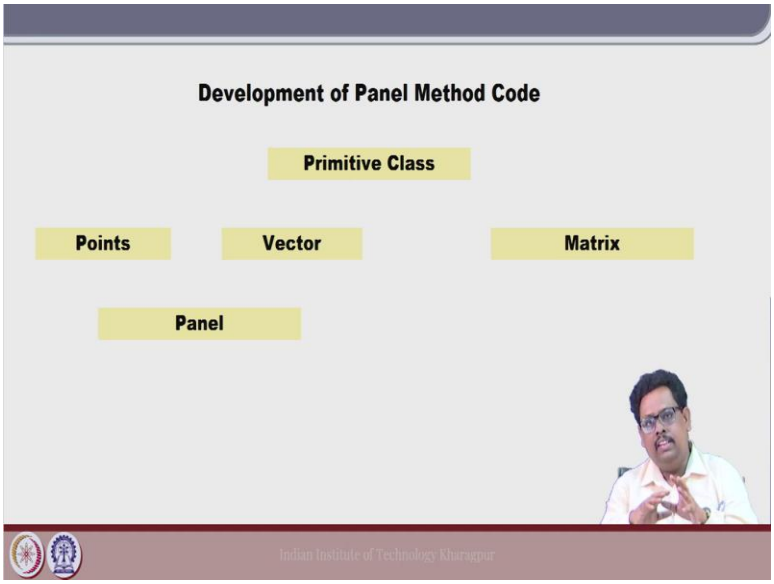


Hello, welcome to Numerical Ship and Offshore Hydrodynamics. So, today we are going to demonstrate a code which is written in C sharp and also we have a code also is written in MATLAB. We discuss if time permit discuss both. Otherwise, today we are going to demonstrate with respect to the C sharp code because it is much more structured way it is written, ok.

(Refer Slide Time: 00:46)



(Refer Slide Time: 00:54)



So, this is the keyword that we are going to use to get this lecture. So, let us discuss this. Now as I discussed last class that there will be something called the primitive class. Now in primitive class what we are going to do is, in primitive class we normally write some part of the structure or some part of the code which is very basic.

For example, if I try to multiply two points then that function may be written in the primitive class. May be suppose you want to make some array from the let us say bigger number to smaller number or smaller number to bigger number. So, these are very

primitive type of you know functions or subroutines or that normally people use for many cases, ok.

So, or maybe let us say the find out the length between the distance between two points that is again a function that belongs to the primitive class. Because distance between two points can be used in finding out many things, right. Or maybe the area of a quadrilateral that also be with may be the part of a primitive classes.

So, these are the functions that we are going to write in the primitive class, ok. The functions can be used in general in many ways, not necessarily particularly solving this problem, ok. So, let us see how we actually structured the primitive classes. Now, to solve this radiation problem in infinite domain I use one primitive class which is called the points. Now you understand that points are a very general thing.

Now, a geometry is nothing, but the collection of points. So, I must have a class which is deals with the points. Now what is there vector? Now vector also you understand it is again very basic thing that you need because let us say try to find out a area of a quadrilateral panel. We discussed that you define a vector between x1 and x2, two points and again you define a vector between x1 and x3, two points and then you do the cross product and then it then you make it half of the cross product.

Again, you take two vector along x 1 and x2 and x1 and x3, right. And then again you can make the same thing and then you add these two triangular area, you will get the area of a quadrilateral. So, I know that for that you need the vector notation. So, also if you try to find out the normal vector.

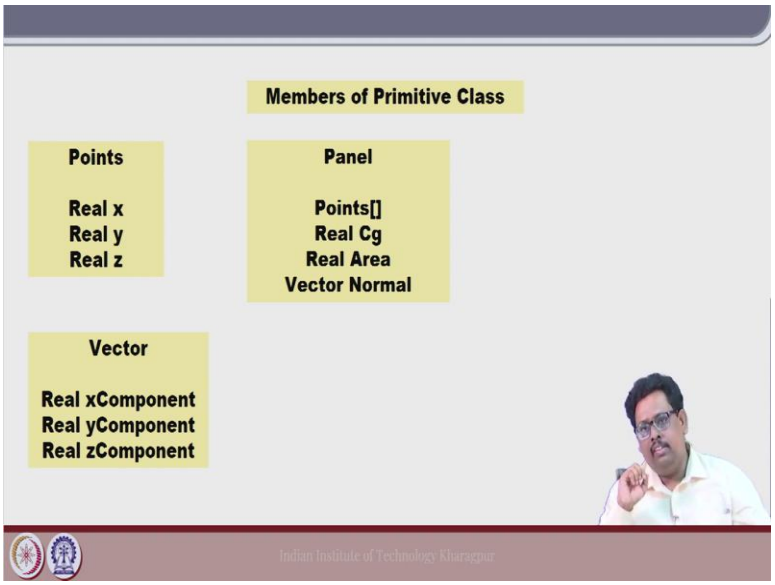
So, therefore, you need definitely a vector, ok. Third one is it is the panel which is not that primitive class, but for attacking this particular problem definitely this panel class or the meshing we can call mesh also in CFD a classical CFD or range based solution we are using it mesh structured un structured.

So, this you can say the panel is nothing, but a mesh. So, why it is required? Because you know that if you want to solve this problem using the panels, right. So, therefore, you have to have a class name called panel, right. And finally, the basic class that I write here is the matrix. Definitely you are going to use the matrix. Why? Because if you remember

that finally, you are having a algebraic equation a x equal to b, right. So, then in order to solve this equation you must have some matrix operation is required.

So, therefore, you need a class, right. Which is you know to help out to do this (A^{-1}) to multiplication between two matrix, addition of two matrix, subtraction of two matrix all this operation you need to store somewhere you need to write somewhere. So, that is why we have this matrix class, ok. So, these are the basic primitive class that I use to solve this particular problem, ok.

(Refer Slide Time: 05:25)



Next like what are the members of the primitive class we need to understand. Now if you look at the point. So, point is the x coordinate y coordinate and a z coordinate. So, therefore, and that is why we call these classes are the primitive class. Because member of this class basically either real number or integer. It is again, we really do not use any such sophisticated structured member in the primitive classes, ok.

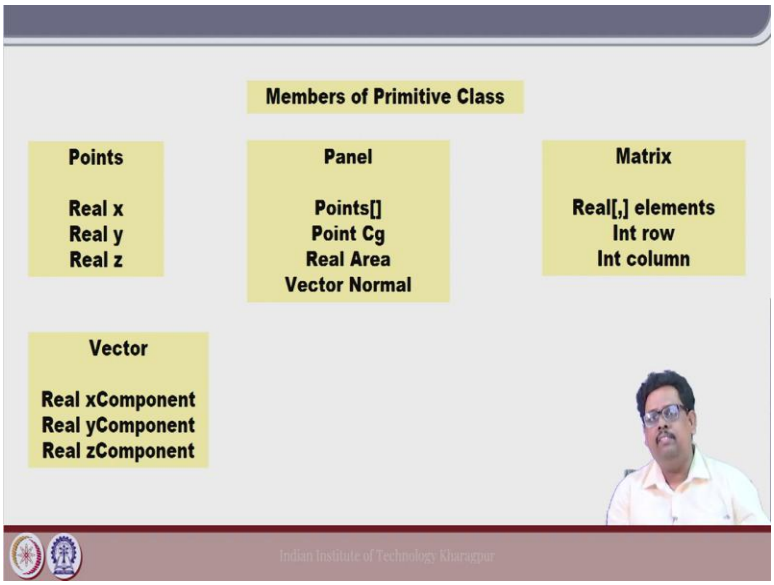
So, these are the members for the points definitely. Now again the vector definitely it is a - it is again a real number which is the x component of a vector, then y component of the vector and the z component of the vector. So, this is the member of the class vector. Now this is more complicated class I would say compared to the points and the vector, but yet it is a simple structure.

And inside the panel that what are the member of the panel; that means, what are the information you are going to store in this panel. First you need a list of points list of points is nothing, but the 4 corner of the panels, right. This either is a quadrilateral panel or its a triangular panel. If it is a quadrilateral panel then it requires the 3 4 points if it is a triangular panel it requires 3 point.

So, the collection of points or the list of points you can say like I do not use the word list though I used in the code is a list of points. However, those who is not very much familiar with C^{++} or C sharp they might not know what is what is the list function ok; however, for them they do understand what is the array.

So, it's a array of points. Then I store one information about the center centroid, right. Definitely because you place your the ϕ at the centroid. So, location of the ϕ is very important, right.

(Refer Slide Time: 07:50)



So, now here so, in panel we have the points we discuss I mean the set of points which is the coordinate and also the centroid, ok. So, its not a real number everyone understand it, right. So, its a type of I just correct it right now. So, its a point. And then you have the area of the panel. So, that should be a real number that is fine yet.

So, we have stored that area also because if you remember that in case of a that one point gauss quadrature rule you need to find out the functional value at the centroid. So, you

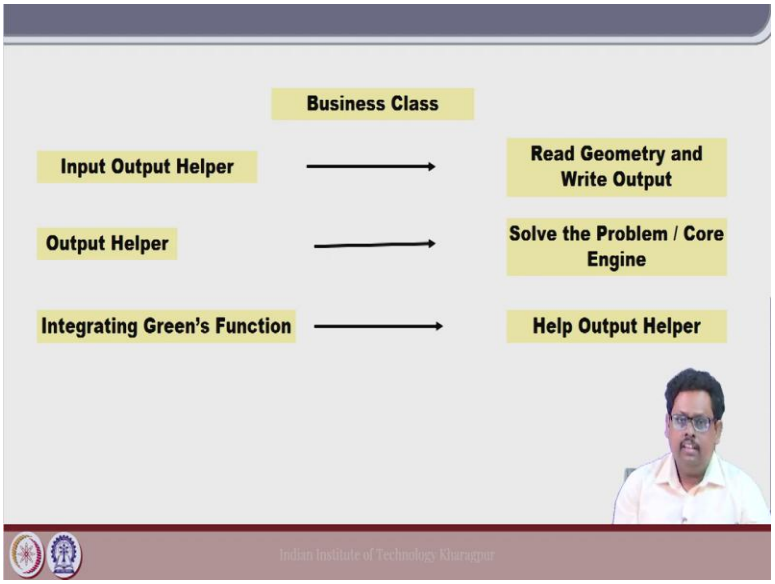
need a location of centroid and multiply by the area of that particular panel. So, therefore, in this panel structure you must have the area. And finally, definitely you need a vector which is nothing, but the normal vector.

Why? Because remember that influence coefficient is nothing, but $r.n\frac{1}{r^3}$. Now in order to get the n you need the panel normal. So, these are the - this is the member of the panels. And finally, the matrix. So, the member of the matrix is nothing, but the array of elements right you have the a (i,j) matrix. So, you need the elements also the row and the column definitely that is how you can write the matrix.

And also, here I am not you know discussing about because the time is limited, we are not discussing about the functions I stored in all these classes. Because in vector for example, inside the vector class, I need some basic functions to write to be used by the user for various occasions. For example, just give some example that dot product of two vector, cross product of two vector and then finding out the unit normal vector, finding out the just normal vector.

So, all these things should be written in vector even the matrix they are you need to inverse of a matrix should be there. That multiplication to matrix should be there, addition of to matrix should be there, subtraction of to matrix should be there, right. So, and also the identity matrix like how to get it identity matrix write a identity matrix just like that. So, this type of function should be written inside this classes also apart from this primitive class what more is required is the business class.

(Refer Slide Time: 10:31)



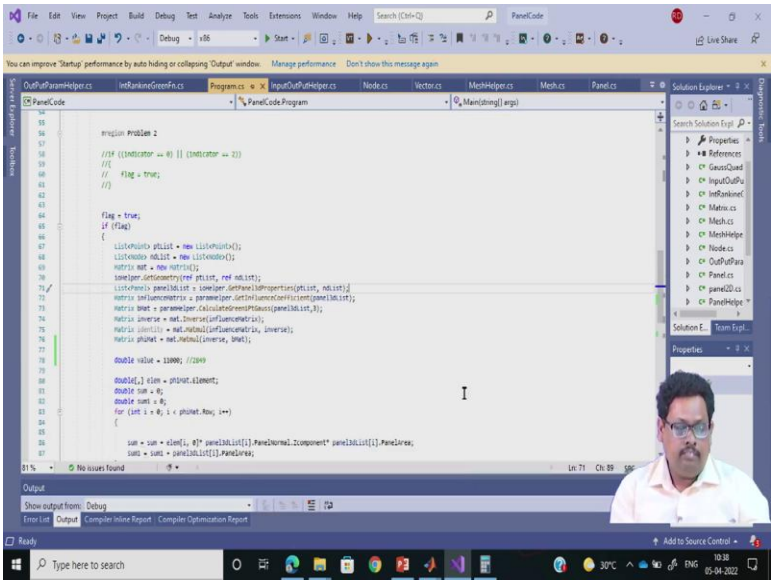
Now, suppose you need to read the geometry and you need to write the output. So, you need the input output helper for to do that, right. So, therefore, this class is not the basic primitive class this class mostly you know you have to deal the business things.

For example, I am reading the geometry I am writing the output. So, for that I have one class which is the input output helper. Similarly, something called the output helper also let us say, sometimes you can have this integrating the greens function also you need some classes, right. So, these are not the basic thing it is particularly these are the required to write the code, I mean to solve this particular problem.

Now, in output folder what actually I write, I just you can call that is the core engine also sometimes, because all this business logic to set up this influence matrix is written in the output helper, ok. Now so, I write the influence matrix inside this output helper I write the right hand side b matrix inside the output helper; however, to get this information that the a(i,j) for example, I need a $r.n\frac{1}{r^3}$.

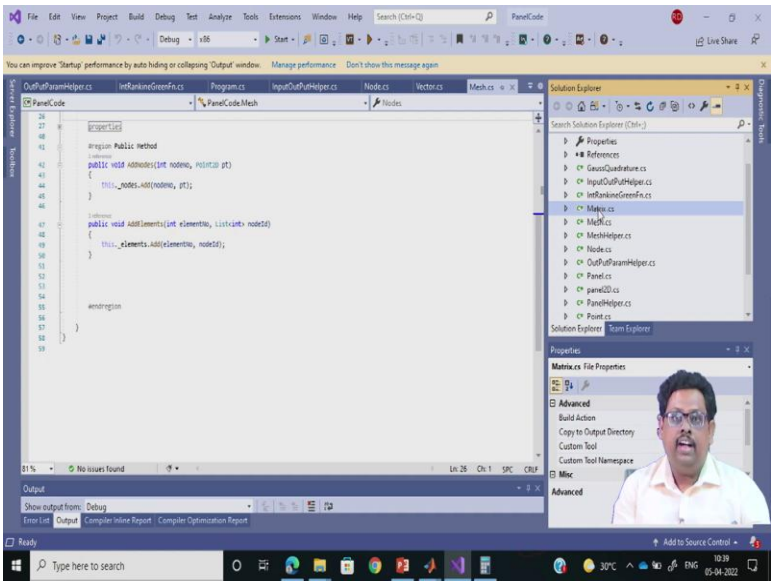
So, this integration is required and for this integration purpose actually I am having the class which is it is which help the output helper. In fact, ok so, the name of the class is not integrating greens function the name is some something some other names is there in original code; however, just to make you understandable. So, I am just mentioning this, ok. So, now, let us with this understanding let us try to see the code, ok.

(Refer Slide Time: 12:38)



Now, this is the code so, just try to make it little bit bigger just I think it is with bigger you can understand it. So, here I do not have anything. So, I just leave this. So, I can make this way, ok.

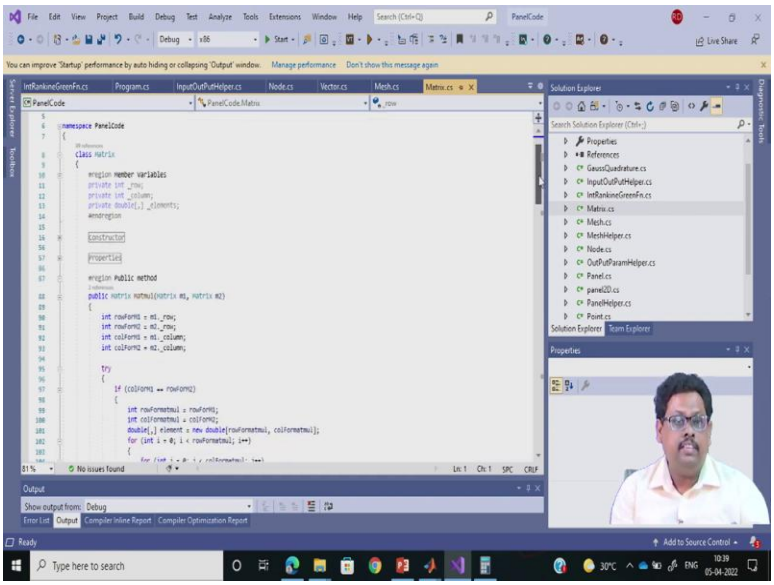
(Refer Slide Time: 13:28)



Now, here if you look at this the top part here first let us see this part you can see here, all this see here actually there are many more classes. Because normally one point gauss quadrature does not work much, we have to go for the higher gauss quadrature rule. So,

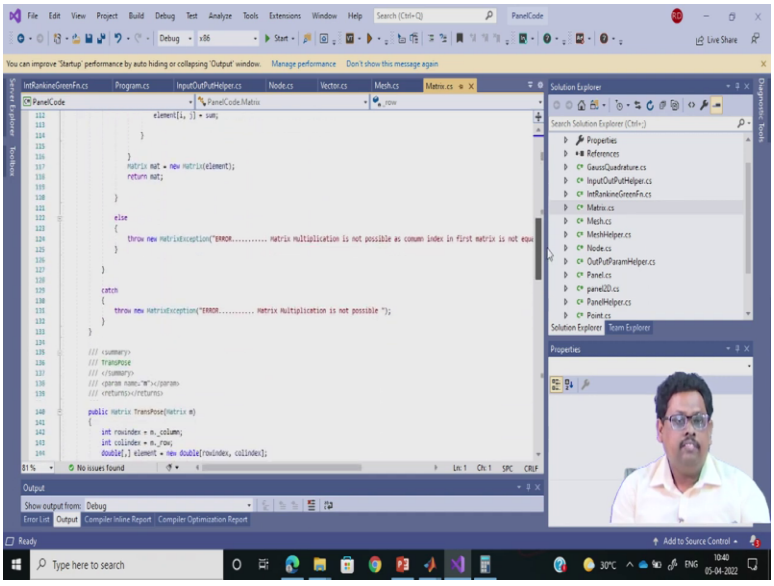
other classes are also there, but this important class all everything is here you can see here the matrix class.

(Refer Slide Time: 13:55)

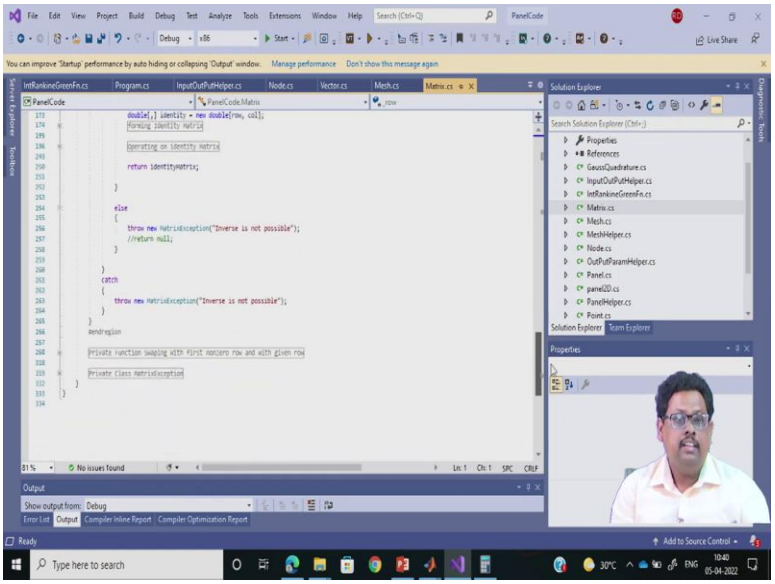


Now, if you click at the matrix class you can see here that member variable of this matrix class. Now, if you look at it is it has row, column and the element as I said. And then you can write this public method that the typical function is written over here.

(Refer Slide Time: 14:22)

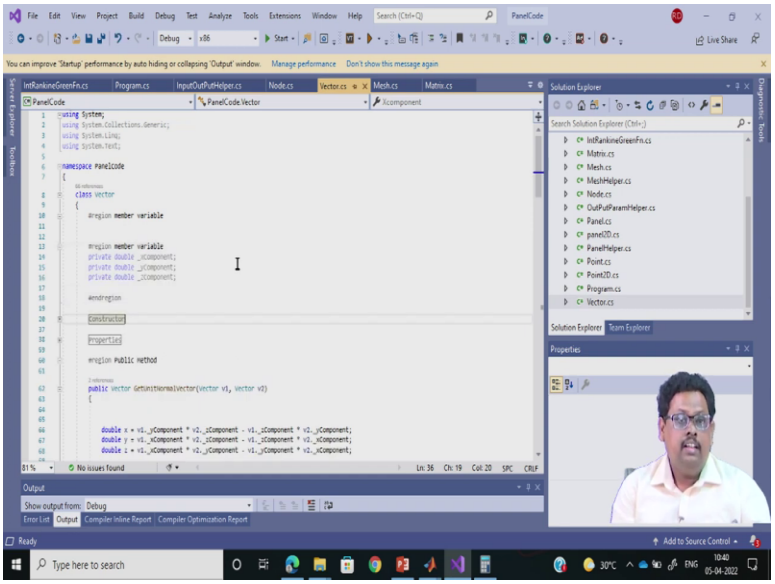


(Refer Slide Time: 14:30)



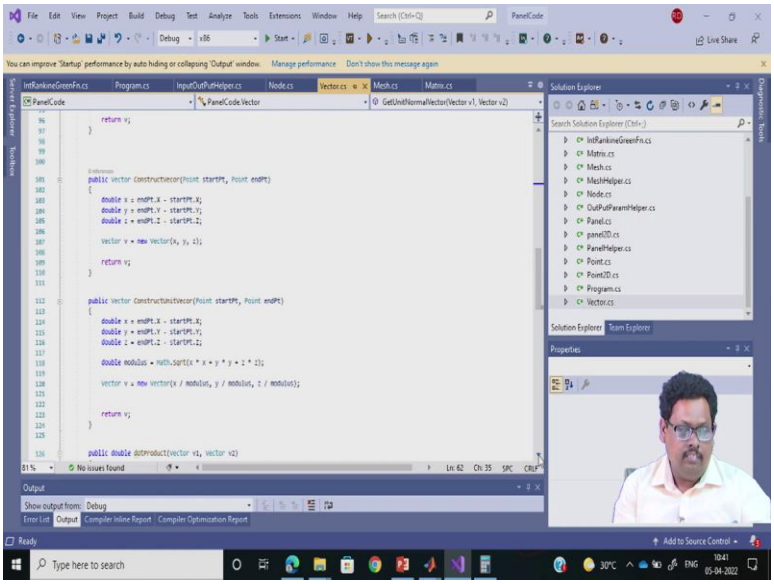
Now, you can see that matrix multiplication with the first function and then it. So, this is one and then you have the transpose of a matrix and then you have the inverse of a matrix. So, everything is written in this the matrix this class. So, let us show let us see the vectors the vector class

(Refer Slide Time: 14:45)



Now, if you look at this the vector class and let us again this member variable again you can see it is this x component y component and the z component, right.

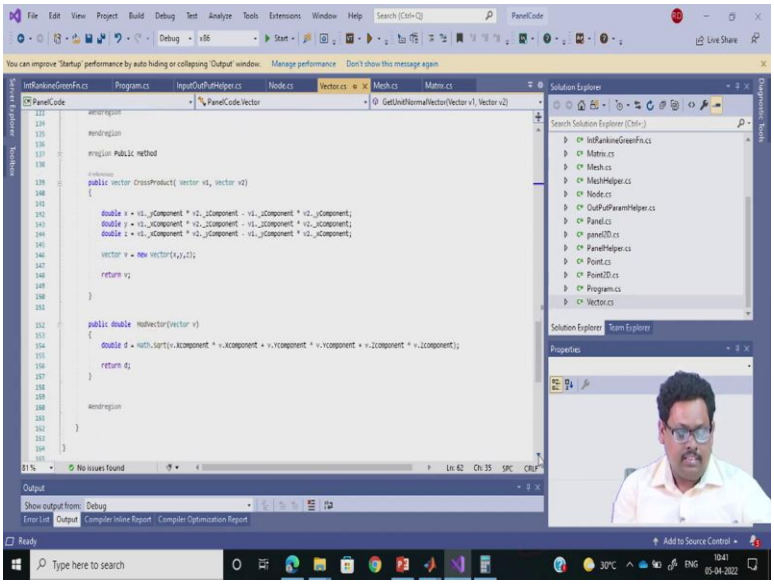
(Refer Slide Time: 15:04)



And all this and then if you look at this the Public Method that is it is basically which the people from other from everywhere they can access it. It is that GetUnitNormalvector then it is get the normal vector and then how to construct a vector.

Suppose if you give the start point and end point then it comes out to be the construction of the vector.

(Refer Slide Time: 15:38)

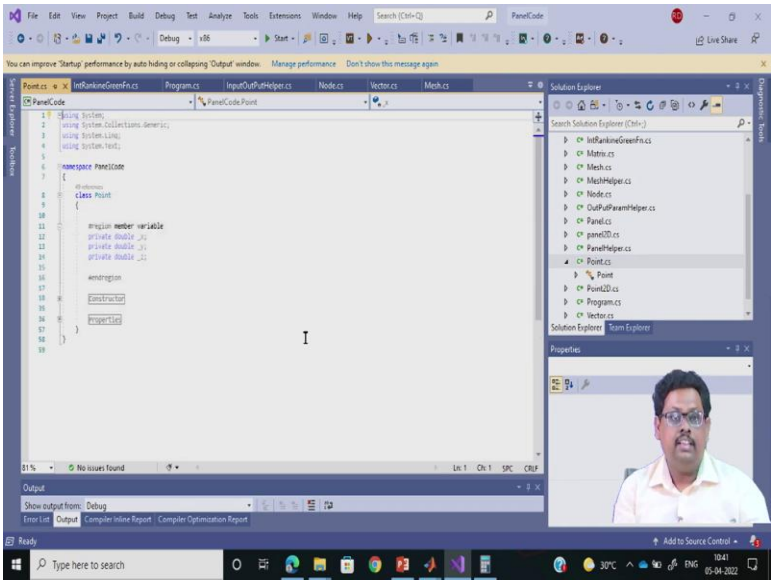


And then the dot product and then the CrossProduct and the modulus. Sometimes actually it I write for particular for this purpose give a vectors and then the return back

the modulus of this vector. Because if you remember the panel area is half of modulus of this, ok fine.

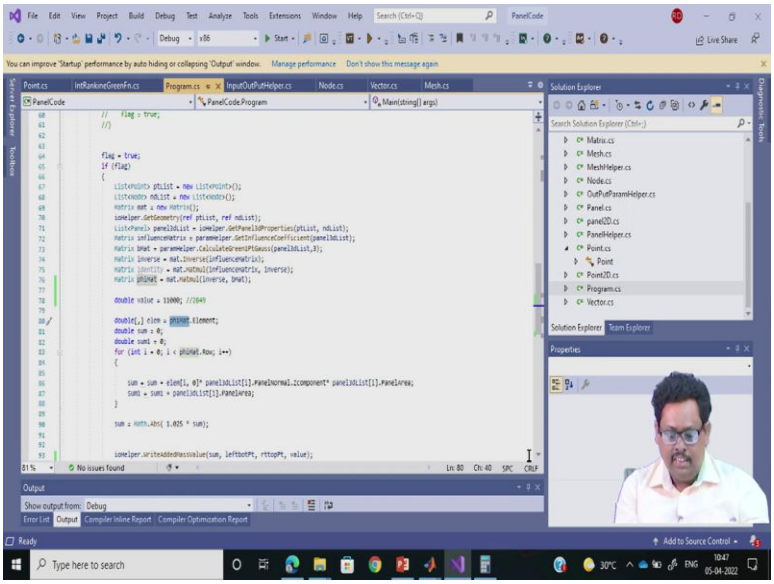
So, these are the, the primitive classes another primitive class is Point.

(Refer Slide Time: 16:06)



Now, again if you see these points it is only has three member, which is the x one y one and the z, I mean x y z three things, right. Ok. So, now, let us see that how this program works.

(Refer Slide Time: 16:23)



So, I this is the main the code program dot cs, where actually I find out the added mass. Now you see here initially I am creating a list of points which is the nothing, but the points from the that is I used I mean that define the geometry of the of the body.

Second one is it is the list of nodes if you remember that point is associated with the node, right. Let us say point x1 I mean one p1, p2, p3, p4 if this four point makes one panel. Then in the node list it will be written is 1,2,3,4. Let us say some p6 p8 p10, p12 these four points make a make one element or a panel then in node list it is written that 6 7 8 11 something like this, ok.

And then I am using this input output helper like, right. So, I just and I find it out the geometry. So, I send this point I send the node list, ok. So, this is the reference actually in this the getgeometry in this function. So, this is first I let us write the code structure, the systemic core structure. You can see that it is only this much that whole thing is done only in this is called the business logic. So, step by step.

So, this getgeometry means I am getting the geometry. And then it is something is a now I am creating. So, in this getgeometry I am getting the list of the points and then the list of the nodes. So, it is my input. So, I am reading the input from the getgeometry that function and again you can see the next one again input output helper getpanel3D properties.

So, in this panel 3D property, let us I get all the panel property, what is the panel property? So, if you remember in panel that all the member variable actually, I am calculating here. So, what is the panel normal? What is the panel centroid? What is the coordinate of the four panels? And what are these? These are the elements and the panel area.

So, these are the thing actually I am getting in this function, ok. Now you see that this is how we write the business logic then I write the influenceMatrix, right. So, once I know the panel and then I write the input the influenceMatrix, right. And this influenceMatrix it is written in the parameter which is you can see it is the output param for help.

That means, the as I said that this is the main core engine that helper where I am writing the influence matrix. And also, I am writing the right hand side the Bmat or B matrix that

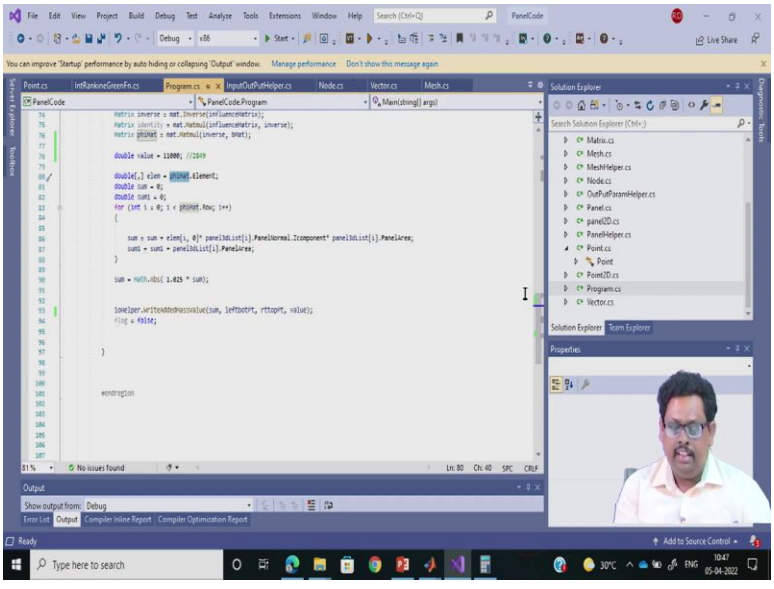
is the right hand side matrix, right which is $\int \frac{1}{r} nds$. That is I am writing over and then you can see that I am now I have to do the matrix influenceMatrix is there. So, I need to do $A^{-1}(B)$.

So, therefore, I am. So, I am calling this I just invert that influenceMatrix. So, I am doing the A is the influenceMatrix. So, here I am finding out the a inverse and then very simple this I just find the it is the check, ok. So, it is the check is that I I try to find out whether if I multiply the influenceMatrix with the inverse matrix. So, A^{-1}=1. So, this is the check this check will you know tells me this yes, your inverse program is correct.

So, once you do that and then I am using the matrix multiplication a inverse into b, I am getting the phi. So, now, when I getting the phi and this is the value that I already calculated for the body, I try to find out the added mass or is the mass of the body because here I did not calculate programmatically what is the mass. So, its the input as I take this as the input and then phi matrix, I extract all the element.

And if you remember that added mass definition is a_i sorry $\phi_i.n_j.ds$, right. So, you need to up find out this. Now in this case my $(\phi_i)_3$, because I am trying to calculate that a33. So, i also 3 and then j also 3. So, therefore, my added mass is nothing, but $\rho\phi_3.n_3.ds$

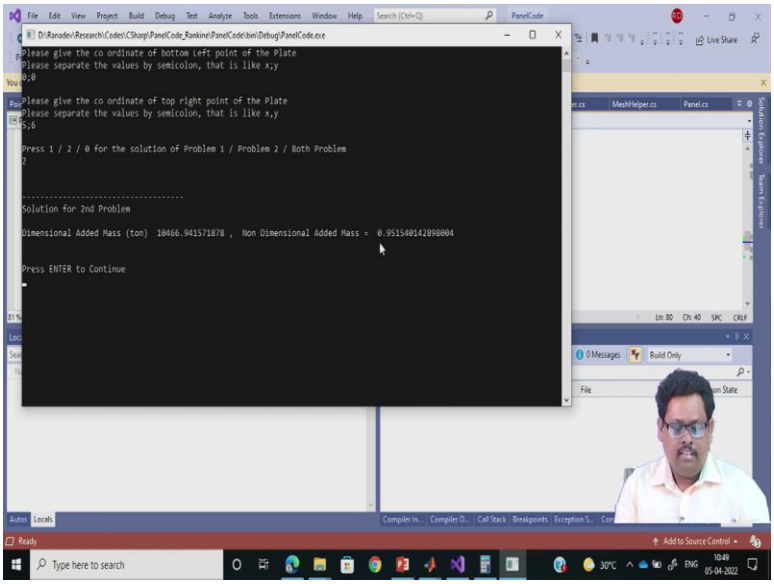
(Refer Slide Time: 21:25)



So, this operation here I am doing you can see here, it is in the sum I get the element and then this I take the PanelNormal the z component which is the n3. And then in this panel I just multiply by the PanelArea, right. And then and then element i 0 is nothing, but the element of the phi.

So, I doing. So, here I am doing phi I multiply by the PanelNormal multiply by the PanelArea. Now if you keep adding everything. So, you will get the total this thing, ok. So, now, with this let us now. So, here I am doing it for I take a barge type of structure ok, and just let us run it. So, I just run this code.

(Refer Slide Time: 22:50)



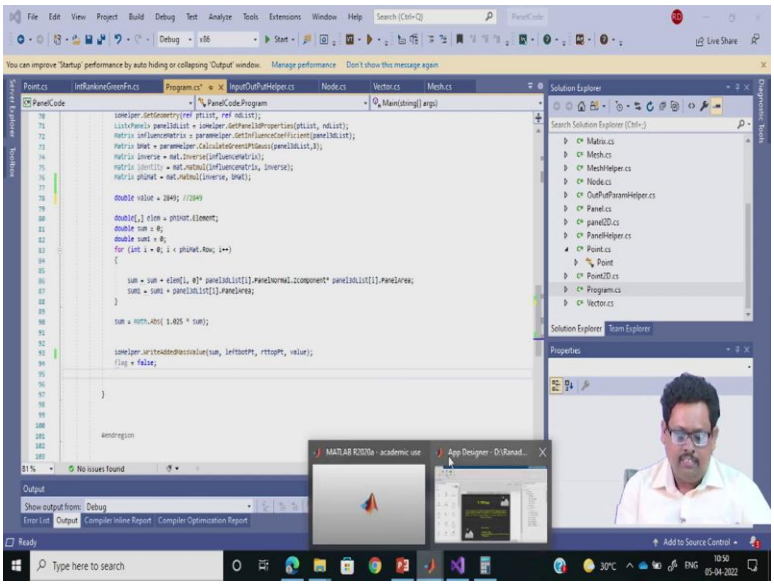
See initially this is for some another problem. So, it is not connected to this here. So, I just press 2 because 2 is the radiation problem. See this why I ask this is for the several in later stage, when you calculate the gauss quadrature rules and try to find out the over the element if the calculation *1/r* is correct or not.

So, for that actually is for the check. So, it is. So, we will discuss this problem number 1 later on. So, let us I solve the problem number 2 first. Now if you do this you can see that added mass the non dimensional added mass becomes 0.95 you can call this as a infinite added mass also.

And in later stage when you actually do the time domain panel method using impulse response function that time this infinite added was also required, ok. But that time we

can found there is another way also finding out this infinite added mass, but this is one way you can find out the infinite added mass, ok fine.

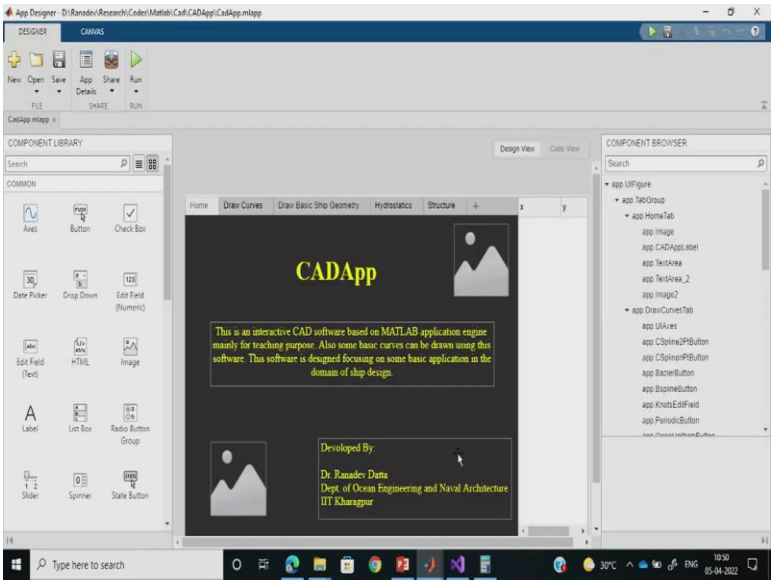
(Refer Slide Time: 24:14)



So in fact, I have taken one more one more case which is the wiggly hull. Now this wiggly hull I can show you that you know now here this getting the barge structure is very easy.

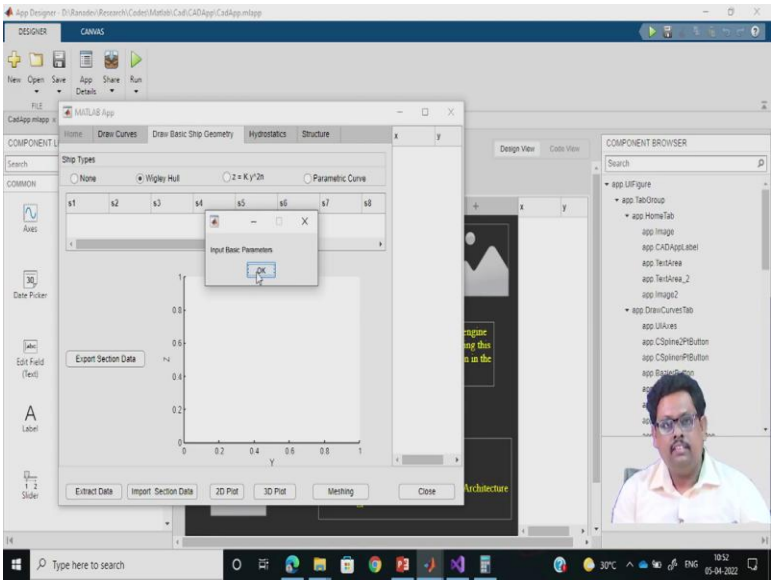
And then if you will get the ship as I mentioned over here to you. That if the panels are given sorry if the sectional lines are given and then how we can make the panel. I discussed in the last class right, you take this next point this, then this point and then that point and then you can do that. So, I did that also here I had our own.

(Refer Slide Time: 24:50)



So, this maybe we can you can use for or you can share in fact, if possible. I do not know we have to see that, but now if I run it, I mean if it is the here what is given a that you can create the section lines and then from there you can create this code will helps you to create the mesh also so, its bit slow. So, it will take some time.

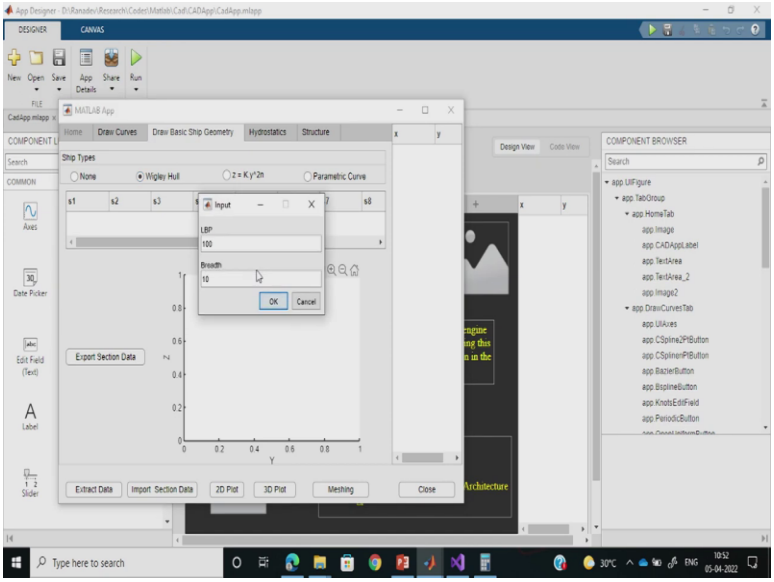
(Refer Slide Time: 25:43)



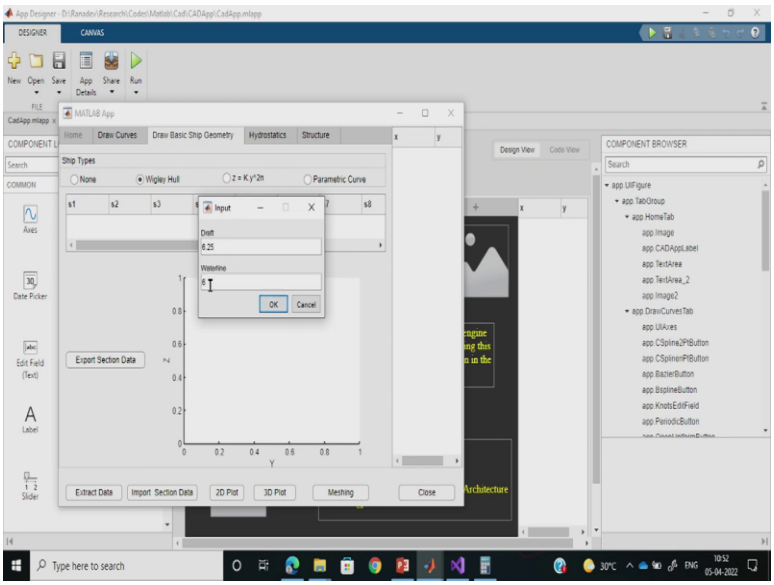
So, I just go this. So, draw some basic geometry. So, this is something called the MBC model that how we can view the thing view the thing. So, it is easier in MATLAB to

visualize things, right. So, I just using this wiggly hull you can see that. So, I just said the basic parameters.

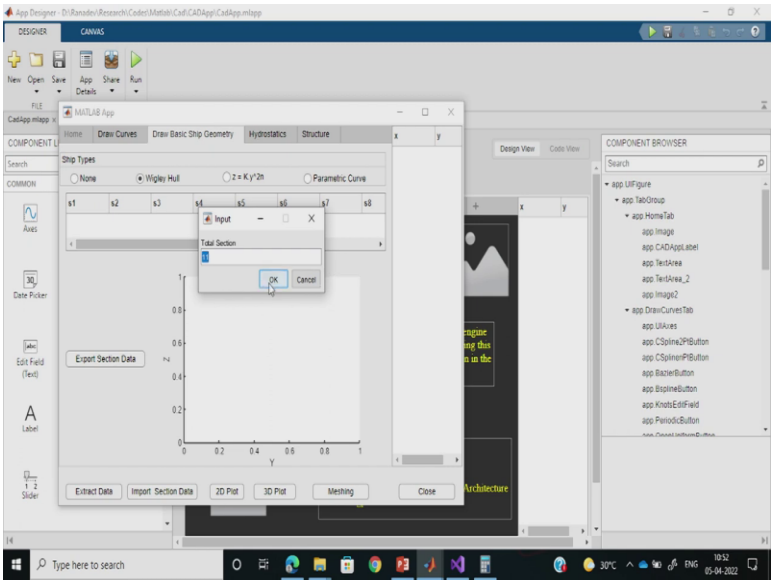
(Refer Slide Time: 26:27)



(Refer Slide Time: 26:40)

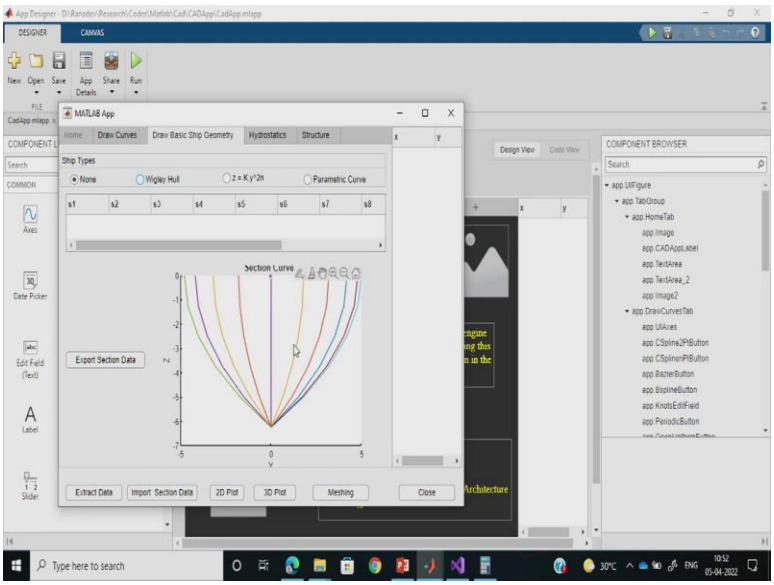


(Refer Slide Time: 26:58)

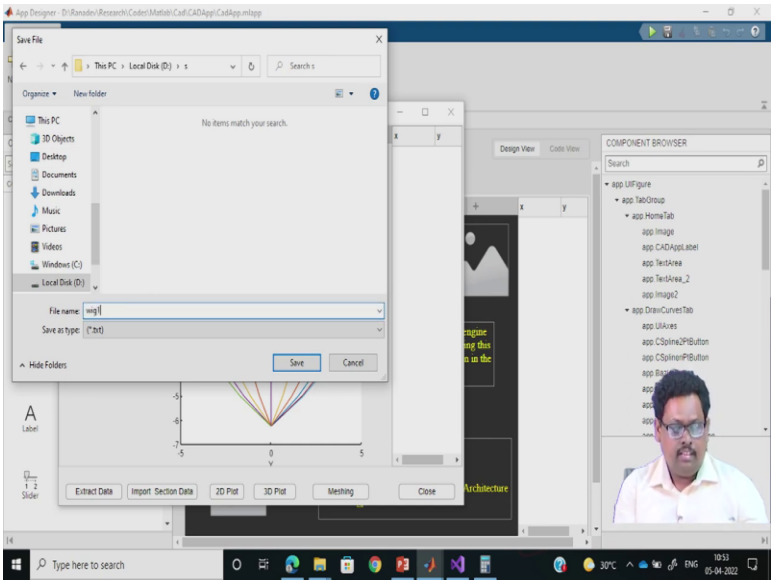


So, I just make it 100 and then breath 0, fine. And let us take because actually I just try to let us give it 6.25, I tell you the reason and then I just make the water line 6 ok, and just total section is 11, ok.

(Refer Slide Time: 27:03)

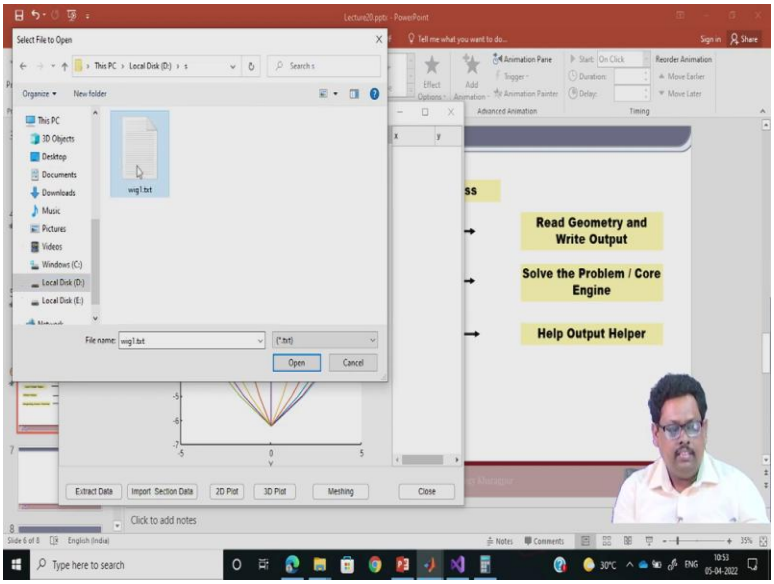


(Refer Slide Time: 27:30)

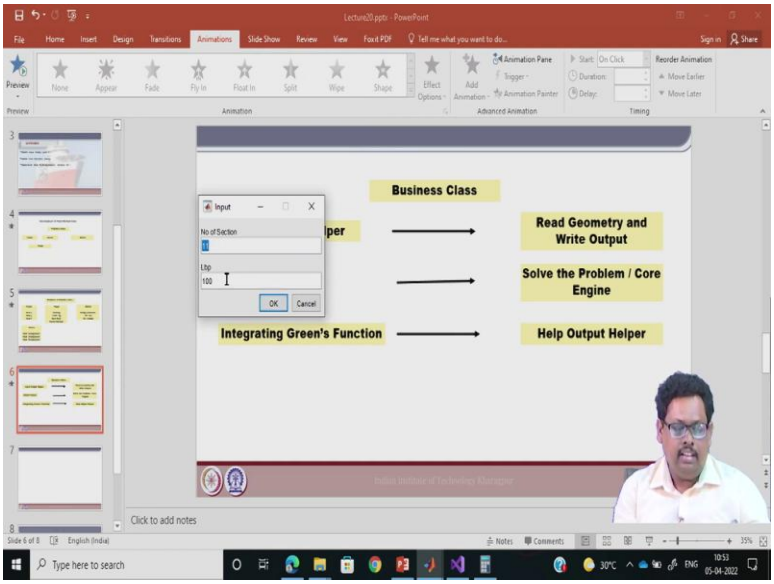


If I do that so, I can get the section lines, right. You can see here I can get the section lines. So, now, I just export this section data somewhere. So, I just save it here. So, I just make it wig1. So, I just save it. So, once you save this data here. Now you can see that it is one thing is given the meshing. So, it just now you have this the section lines data you have. Now you can just the mesh. So, you need to select that which section lines you need to mesh.

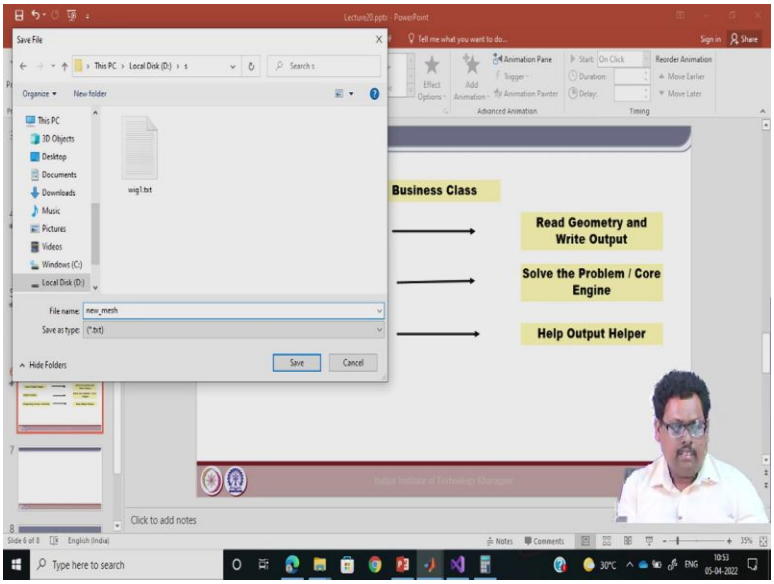
(Refer Slide Time: 27:55)



(Refer Slide Time: 28:01)

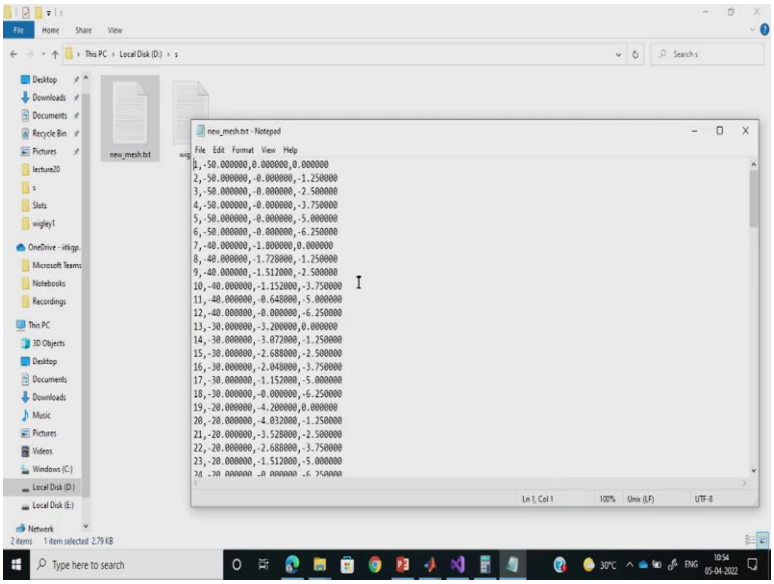


(Refer Slide Time: 28:13)



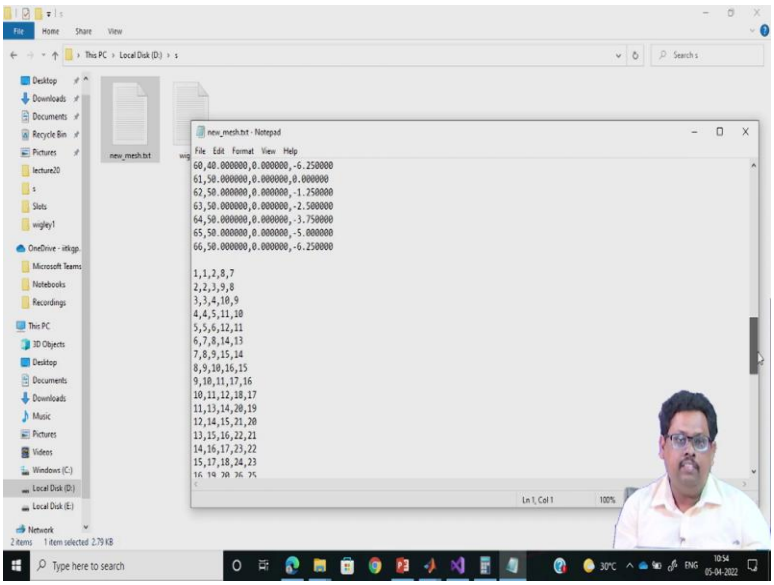
So, I just give it this one and then section lines and Lbp is just cross checking. So, I said yes this is the thing, ok. And then we need to mesh. So, I will just create some new mesh here. So, I just save it. So, just I just show you here actually it is let us see the location. So, you have this new mesh.

(Refer Slide Time: 28:37)



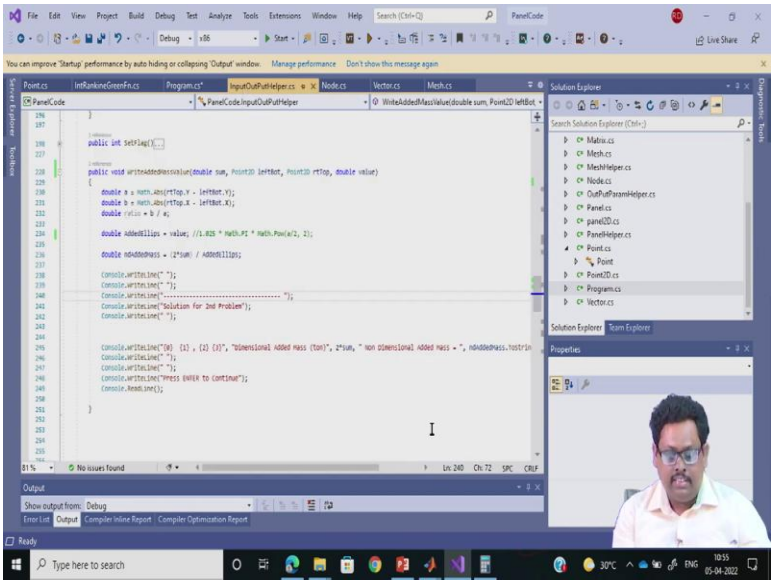
Now, if you look at this the mesh you can see here the first is basically the le all these points is given. And then with a space and this is something called the connectivity you can see that the first connectivity is 1,2,8,7.

(Refer Slide Time: 28:50)



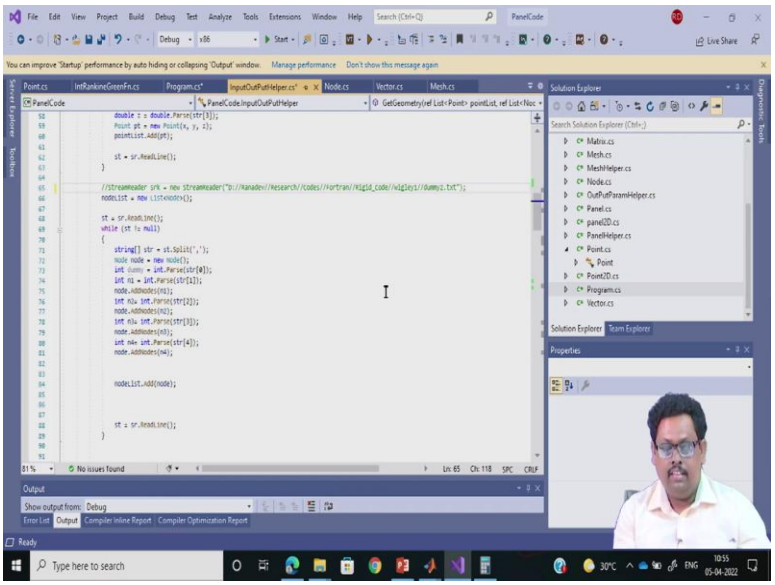
Now, if you look at this the code here 1 is -50, you can take the x and z -50 0, 2 is -50, -1.25 right. And then the last two point is 8 and 7. So, if you look at this 8 it is -40, -1.25, and -40 0. So, you can see that I am making this the anticlockwise notation, right. So, already I have done this and this is already I use over here.

(Refer Slide Time: 29:44)

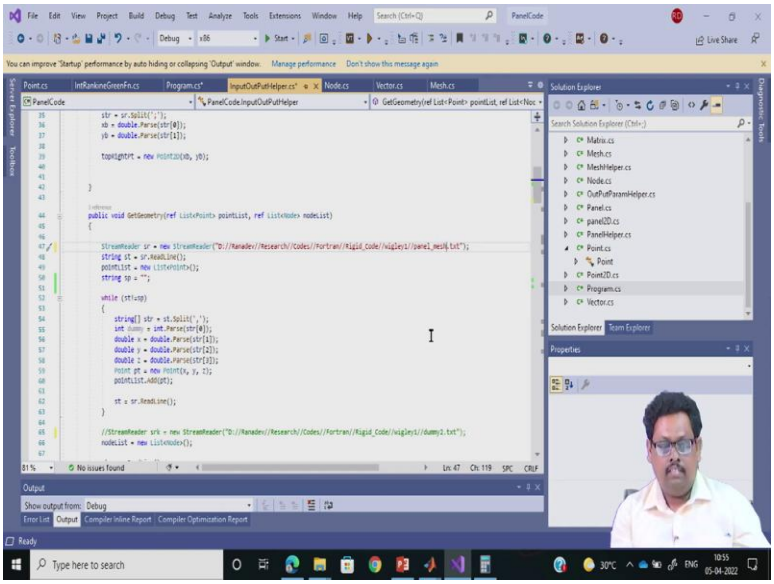


So, in instead of this I can go back to my the original code, which is input output helper and I just change the file name here. Instead of dummy2, I just it is the I think panel mesh like this no not here.

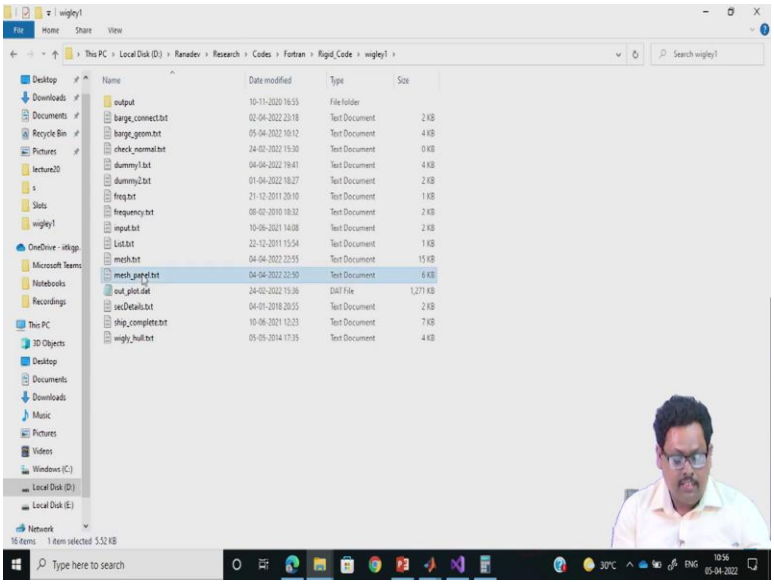
(Refer Slide Time: 30:02)



(Refer Slide Time: 30:12)

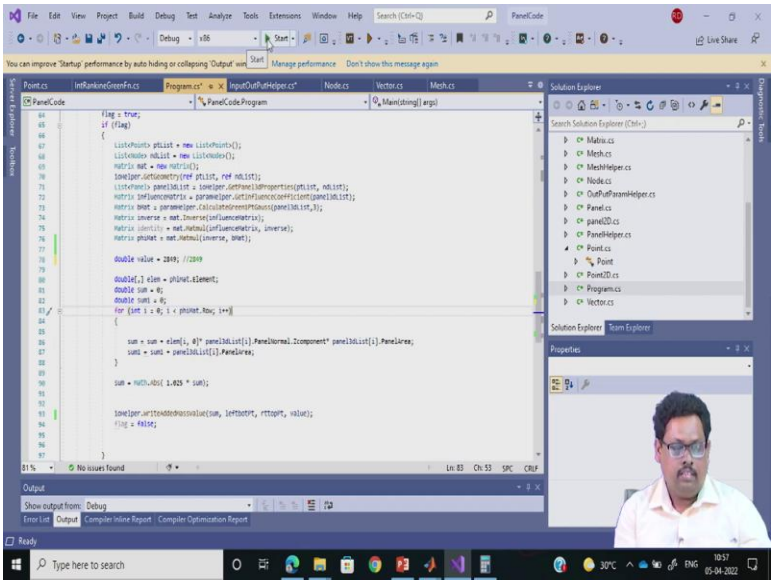


(Refer Slide Time: 30:53)



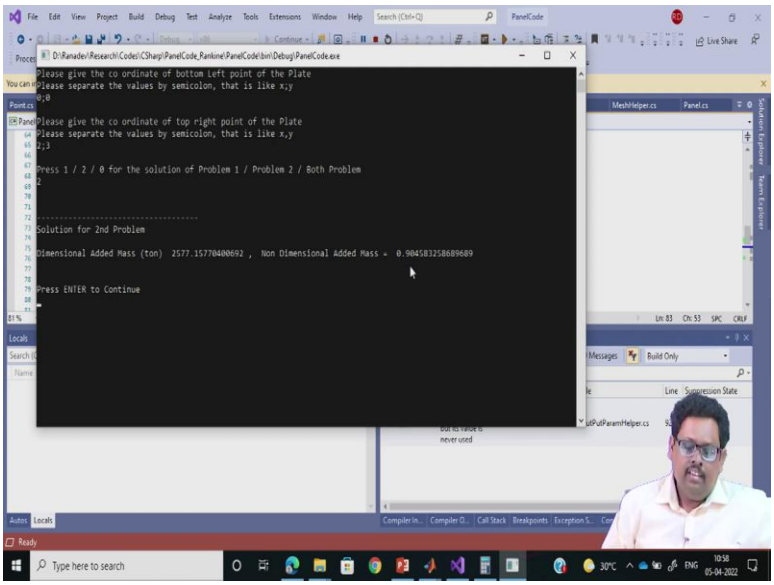
It is the commented line, here I just use the panel mesh. So, just check that if this is it mesh underscore panel, sorry.

(Refer Slide Time: 31:20)



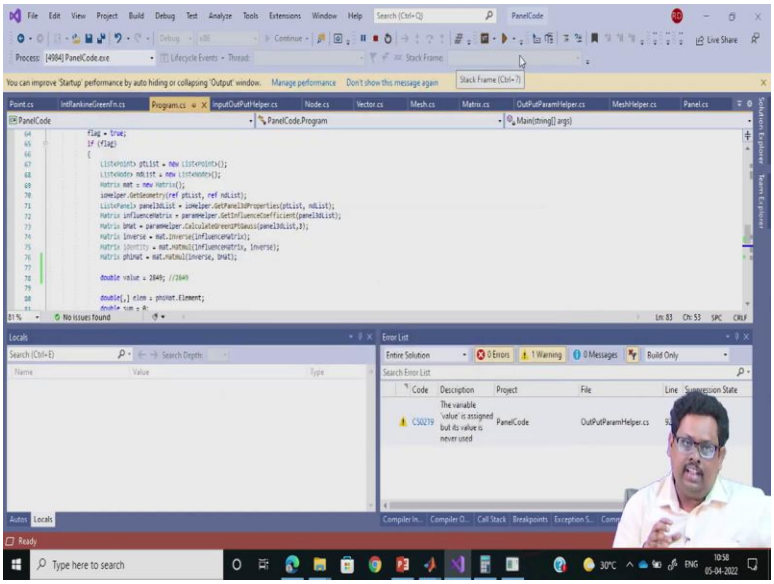
So, now if you and also, I need to change the in the program the mass also because that is I used as a input, ok it is there, ok.

(Refer Slide Time: 31:53)

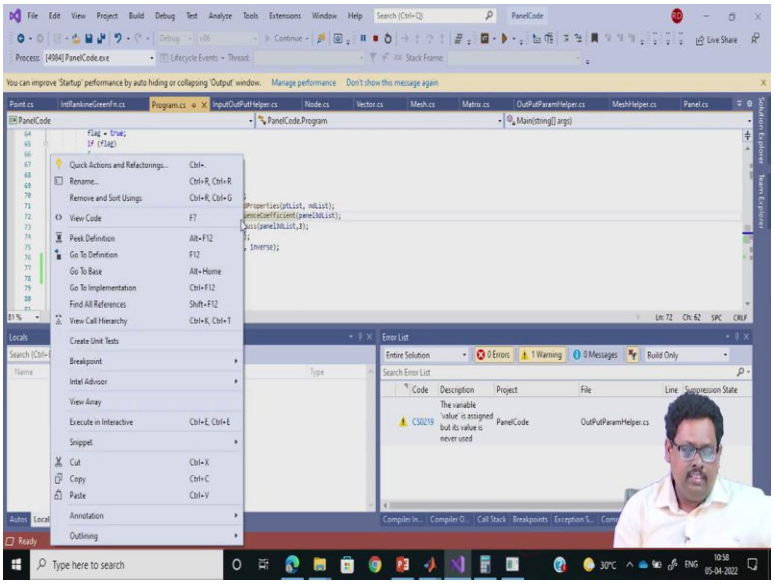


So, this part I can ignore simply just took any junk values. Now you can see here now this added mass is coming at 0.9. So, you see this in case of the ship the barge and here it is more or less 0.9 is coming at the ship added mass, right. So, if you remember that is when you solve the problem that time also whether use this the ship added mass the 90 % of the mass anyways.

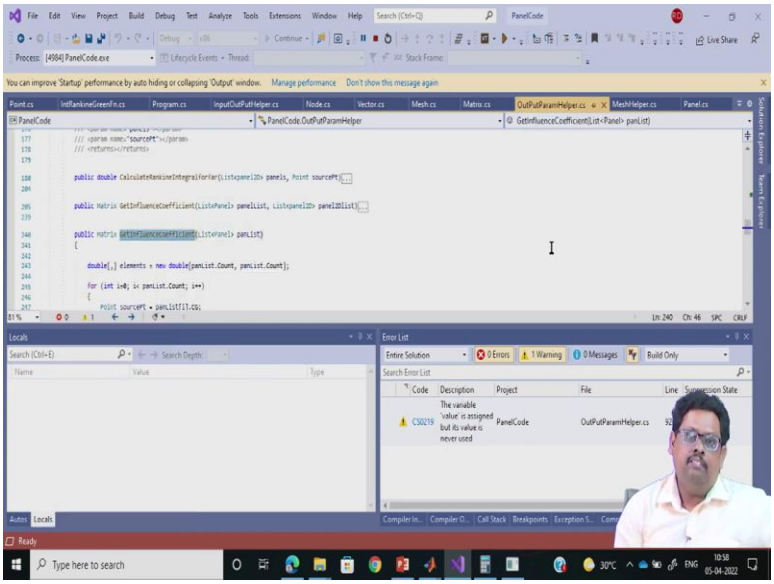
(Refer Slide Time: 32:35)



(Refer Slide Time: 32:51)

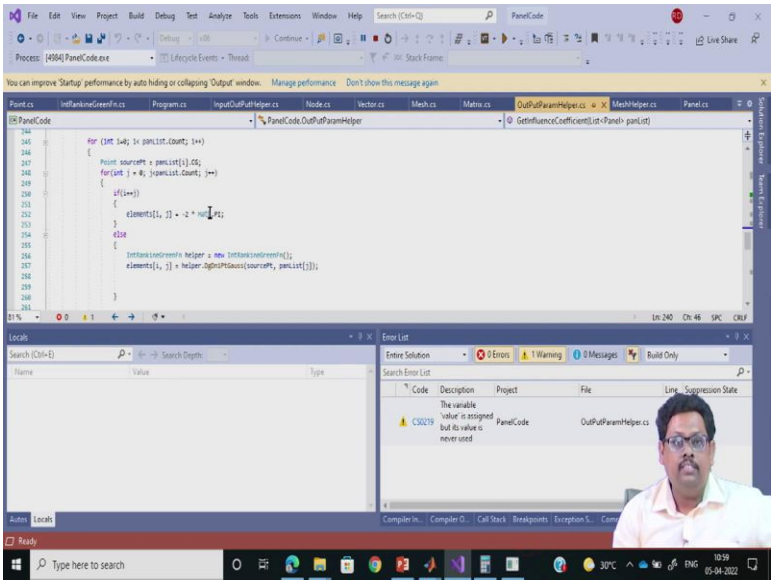


(Refer Slide Time: 32:54)



So, this is how you know this is the way one should write the code. So, you make this code in small business logic and inside this business logic you I mean inside this code you can further do the thing. So, the only the thing let me show you only the influence matrix thing which is which maybe you can see that is a bit more complex remaining thing is very straight forward, right.

(Refer Slide Time: 33:02)

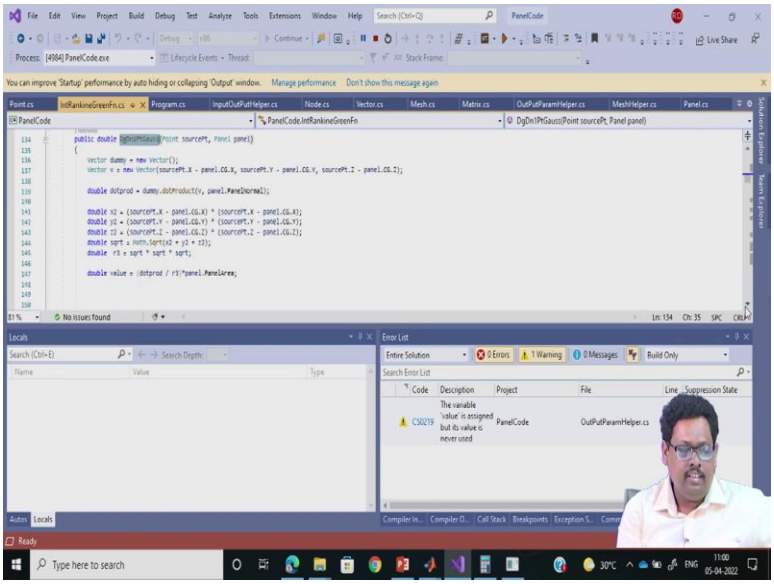


Here see I did nothing here if you remember if you remember this one is the the source point which is the i^{th} panel. So, I write point the source point as the i^{th} panel and I take

the centroid of this particular panel. So, this is how I take this. And if this i = j, I make it is -2π, right.

And again, if you see that if else, I did not calculate here see here you can see the code is very small, that is how you should write the each code the code should you can see this is the that small code. And then here in the right hand side I am using that I call another function which is the $\partial g \partial n$ using 1 point gauss quadrature.

(Refer Slide Time: 33:54)



So, let us go there also. Now if you go here now you can see things are become very simple. Here I have only the source point and then I have a panel. There I have the business logic if i $\neq$ j. So, I am sending you the jth panel, right and I need to do that r_{ij} $\partial g \partial n \cdot \frac{1}{r_{ij}}$.

So, I set the centroid of the i^{th} panel and the whole j^{th} panel because to do the - I need to, because I need the area of the jth panel I need the normal of the jth panel. Now you can see here I am defining a vector v and then I just find it out the r that $\vec{r}$ which is $(x-x\xi)-(y-y\eta)-(z-z\zeta)$. So, that I do over here and then I am using the dot product for the $\vec{r}.n$, right.

So, so I have already defined the $\vec{r}$ which is v. So, I just $v.n$. So, $\vec{r}.n$ I am doing here dot product. So, that is why does this vector things are important. And then I am using the

that you know r^3. So, I am making x = $(x-\xi)(y-\eta)(z-\zeta)$ and then I am making square root of $\sqrt{x}$ of $\sqrt{y}$ of $\sqrt{z}$ and that is my r and then it is r ^{3}.

So, value equal to dot product divided by r^3 and then it should be multiplied by that PanelArea that is how we calculate the influenceMatrix, right. Now, you see that this code also very simple very small. So, that is what I am trying to tell you that write the small small part break the code into as much small as possible and then you integrate all these small small pieces to write this the panel method code.

So, I hope with this lecture after having this 3 session on how to do the coding you can really able to write this panel method code, ok so, yeah. So, next class onwards we are going to continue with the frequency domain panel method, ok.

Thank you very much.