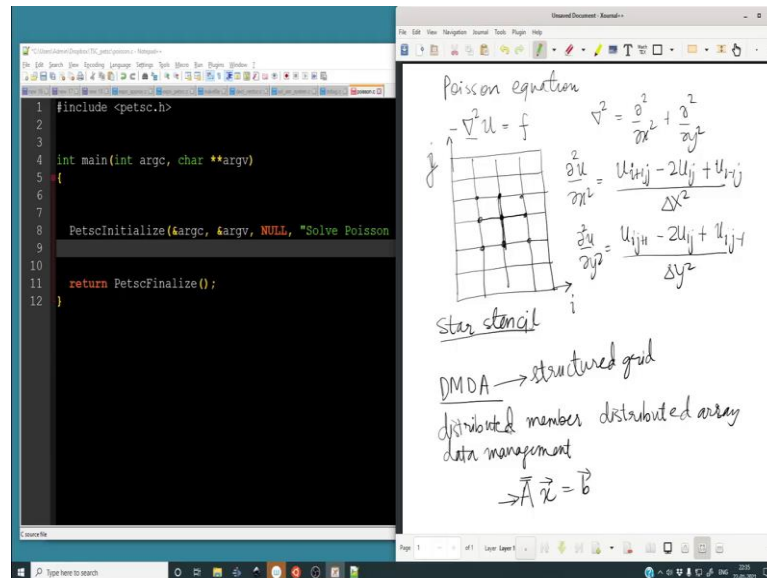


Tools in Scientific Computing
Prof. Aditya Bandopadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 32
Poisson equation in PETSc

(Refer Slide Time: 00:27)



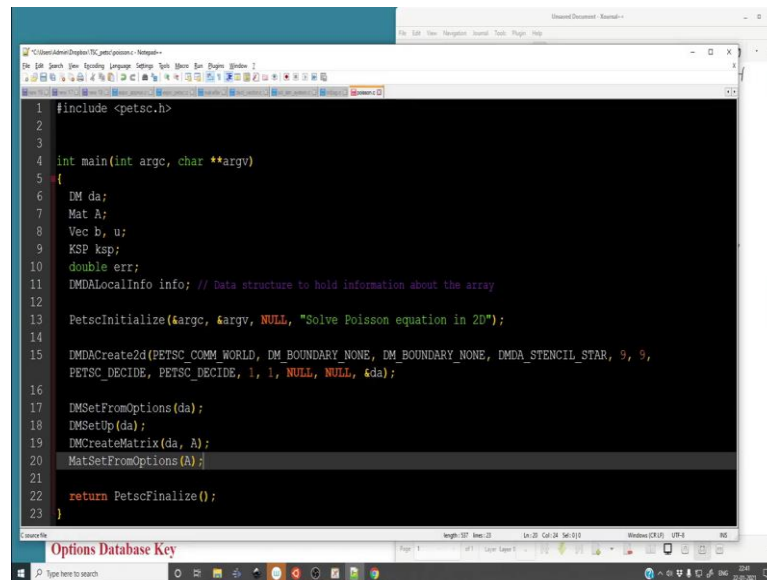
Hello everyone, in the last lecture, we had looked at solving a tri diagonal matrix system. In this particular lecture, we are going to focus on solving a Poisson equation. So, the general form of the equation is the Laplacian of u is equal to some function. And we are going to solve it on a two-dimensional domain.

So, if you recall we had solved this particular problem in Python, and we had made use of a structured grid. We have discretized this Laplacian using central differences. So, just to recap the Laplacian is $\partial^2 / \partial x^2 + \partial^2 / \partial y^2$ and the approximation the finite difference approximation.

So, if this is the i direction and this is the j direction, then at the node i, j $\partial^2 u / \partial x^2$ can be written as $u_{i+1,j} - 2u_{i,j} + u_{i-1,j}$ divided by Δx^2 . Similarly, $\partial^2 u / \partial y^2$ can be written as $u_{i,j+1} - 2u_{i,j} + u_{i,j-1}$ divided by Δy^2 . So, it makes use of these four neighbouring points. So, in the terminology of PETSc this particular stencil is called as a star stencil.

There is one more stencil in PETSc that is called as the as a box stencil, it makes use of these points as well. In this particular case, we will not be needing it. So, we will be focusing on the star stencil. So, let us get started with the program. And as we go through the program, we will discuss the various terms and keywords as and when they appear.

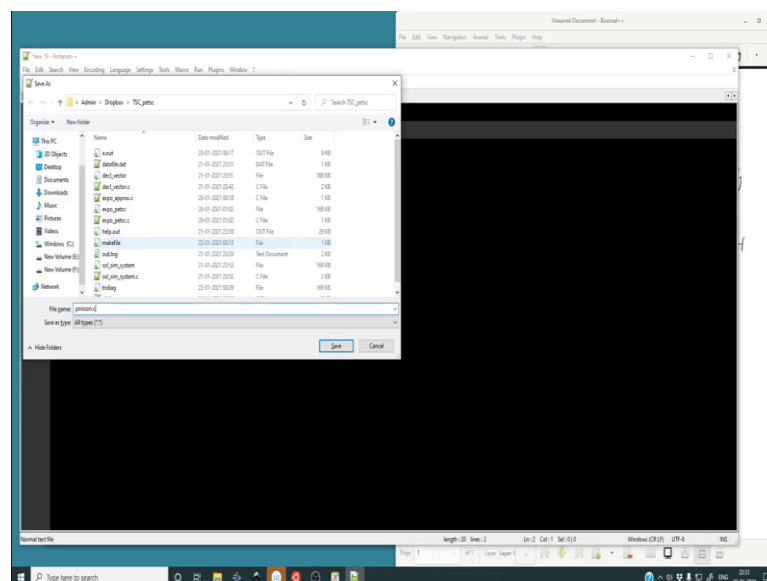
(Refer Slide Time: 03:07)



```
1 #include <petsc.h>
2
3
4 int main(int argc, char **argv)
5 {
6     DM da;
7     Mat A;
8     Vec b, u;
9     KSP ksp;
10    double err;
11    IMDALocalInfo info; // Data structure to hold information about the array
12
13    PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
14
15    IMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, IMDA_STENCIL_STAR, 9, 9,
16    PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL, &da);
17
18    DMSetFromOptions(da);
19    DMSetOp(da);
20    IMCCreateMatrix(da, A);
21    MatSetFromOptions(A);
22
23    return PetscFinalize();
24 }
```

So, let us start. So, first thing is first. We must include petsc.h. Let us save the file.

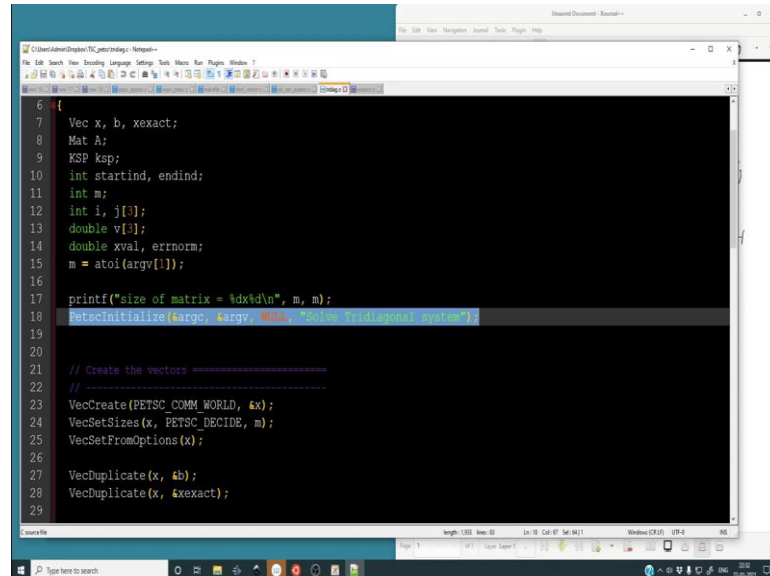
(Refer Slide Time: 03:21)



So, let me call it poisson.c alright. So, then we must obviously, have the main function. So, int main then int argc, char star star argv, this is just to pass command line arguments

alright. So, now we must as usual do two things. One is return PetscFinalize, and we must begin with PetscInitialize alright. So, PetscInitialize as usual takes the arguments.

(Refer Slide Time: 04:15)



```
6 {
7   Vec x, b, xexact;
8   Mat A;
9   KSP ksp;
10  int startind, endind;
11  int m;
12  int i, j[];
13  double v[];
14  double xval, errnorm;
15  m = atoi(argv[1]);
16
17  printf("size of matrix = %dx%d\n", m, m);
18  PetscInitialize(&argc, &argv, 0, "Solve Tridiagonal system");
19
20
21  // Create the vectors
22  //
23  VecCreate(PETSC_COMM_WORLD, &x);
24  VecSetSizes(x, PETSC_DECIDE, m);
25  VecSetFromOptions(x);
26
27  VecDuplicate(x, &b);
28  VecDuplicate(x, &xexact);
29
```

So, we will copy it from our previous program alright. So, this will be solve Poisson equation in 2D alright. So, once we have done this, we must declare something which is called as a DMDA. So, DMDA the data structure, DMDA, so the full form I mean there is no agreement on what this full form should be, but it is good enough if you can think about it as distributed member or I mean data management, I mean there are various things. And the last thing is distributed array.

So, DMDA could stand for any of this, it is not sure what it is, but essentially it is used for a structured grid, as opposed to the case of an unstructured grid that one uses for problems which have a complicated geometries in finite element methods and so on. So, where you have unstructured grids, but here we will focus on structured grid.

So, DMDA is a PETSc data structure, which allows us various I mean not shortcuts, but it helps us in creating the finite difference matrix A, where A times the unknown is equal to b the right hand side. So, it helps us in constructing this without having to work a lot towards it.

So, we must create this DMDA matrix. So, let us do that. So, first we must declare DM as a variable da. Then apart from this we will also need the matrix A, we will need the

vectors b and u , we will also need the solver `ksp`. So, these things we have discussed already. And apart from this, we will need the error. So, `double err ok`.

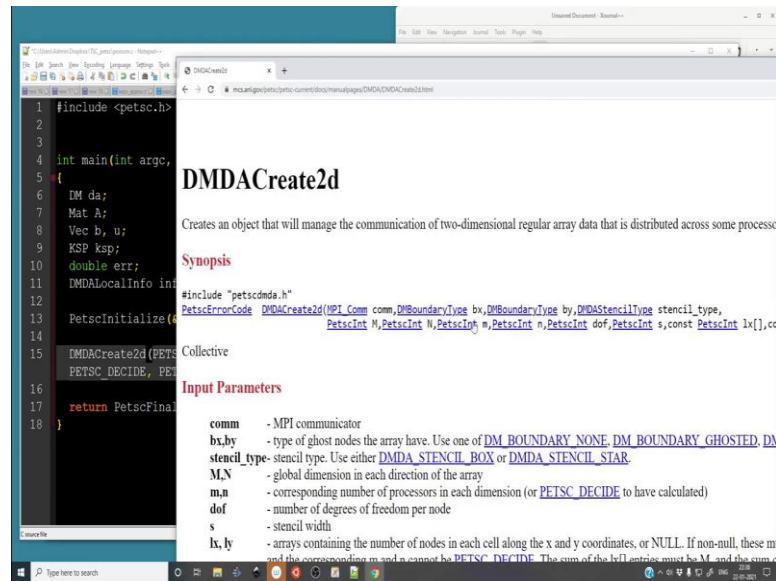
Apart from this one more utility is `DMDALocalInfo`. So, it is a data structure which holds information about the structuration of the grid; in particular, it will tell us the bounds of the grid, and it will help us in performing loops. So, this is a very useful data structure alright. So, now, that we have done this we can go ahead and create the `DMDA` and declare it over the matrix A .

So, `DMDACreate2D`. So, obviously, you can imagine there is a 3D version as well. So, the syntax is first we have to pass the communicator the MPI communicator then we must pass the nature of the boundary. So, in this case the nature of the boundary will be `NONE`. There is no special thing we have to do about the boundary. And if the boundary is periodic, then you must supply `DM boundary periodic`; otherwise it is good enough to have this.

Then we must tell what kind of grid we will have. So, in this case we are going to have a star stencil. So, `DMDA STENCIL STAR`. Then finally, we must. So, we what we can do is declare a default size. So, `9 cross 9`, and finally, we must pass the subdivision of the `9 by 9` grid.

So, like in the previous examples, we will ask PETSc to help us in balance the load. So, it should be `PETSC DECIDE, PETSC DECIDE`. So, it is for both x and y directions then. So, let us, let me open the syntax real quick. It will be clear what is going on ok.

(Refer Slide Time: 09:45)



The screenshot shows a code editor with the following C code snippet:

```
1 #include <petsc.h>
2
3
4 int main(int argc,
5 {
6     DM da;
7     Mat A;
8     Vec b, u;
9     KSP ksp;
10    double err;
11    DMDALocalInfo info;
12
13    PetscInitialize(&argc, &argv, 0, NULL);
14
15    DMDACreate2d(PETSC_COMM_WORLD,
16                PETSC_DECIDE, PETSC_DECIDE,
17                PETSC_DECIDE, PETSC_DECIDE,
18                1, 1, 1, 1, 1, 1,
19                NULL, NULL,
20                &da, &A, &b, &u, &ksp, &err);
```

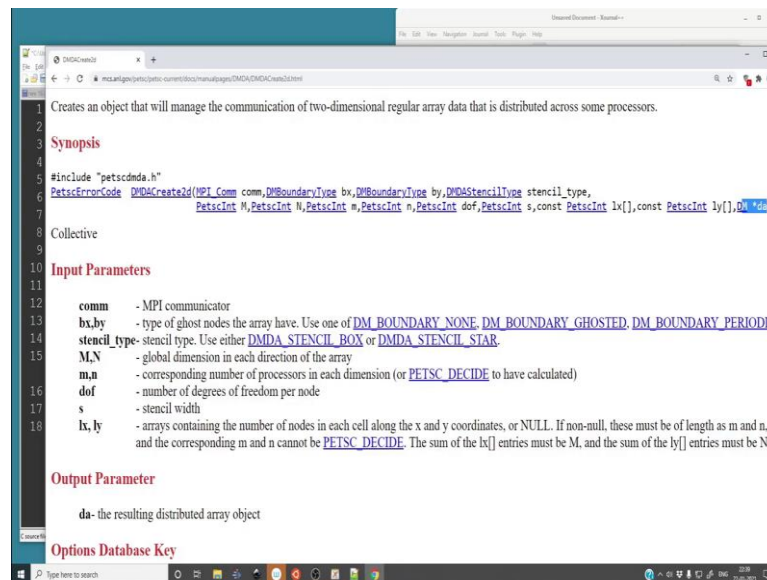
The documentation for **DMDACreate2d** is displayed to the right of the code. It includes a synopsis, a synopsis block with function signature, a collective note, and a list of input parameters:

- Synopsis**
Creates an object that will manage the communication of two-dimensional regular array data that is distributed across some processes.
- Synopsis**
`PetscErrorCode DMDACreate2d(MPI_Comm comm, DMBoundaryType bx, DMBoundaryType by, DMDAStencilType stencil_type, PetscInt M, PetscInt N, PetscInt m, PetscInt n, PetscInt dof, PetscInt s, const PetscInt lx[], const PetscInt ly[], DM *dm, Mat *mat, Vec *vec, KSP *ksp, double *err);`
- Collective**
Collective
- Input Parameters**
 - `comm` - MPI communicator
 - `bx,by` - type of ghost nodes the array have. Use one of `DM_BOUNDARY_NONE`, `DM_BOUNDARY_GHOSTED`, `DM_BOUNDARY_PERIODIC`
 - `stencil_type` - stencil type. Use either `DMDA_STENCIL_BOX` or `DMDA_STENCIL_STAR`
 - `M,N` - global dimension in each direction of the array
 - `m,n` - corresponding number of processors in each dimension (or `PETSC_DECIDE` to have calculated)
 - `dof` - number of degrees of freedom per node
 - `s` - stencil width
 - `lx, ly` - arrays containing the number of nodes in each cell along the x and y coordinates, or NULL. If non-null, these must be the same as the corresponding `m` and `n`, and cannot be `PETSC_DECIDE`. The sum of the `lx` entries must be `M`, and the sum of the `ly` entries must be `N`.

So, we have done this then we have to pass the degree of freedom. So, degree of freedom meaning what is the, so if it is a scalar it has only one degree of freedom, and we have to put this stencil width and so on. So, stencil width will be 1, the degree of freedom will be 1 that is why we have 1, 1. Then we have to say arrays containing number of nodes in each direction.

So, if in general you will simply set it to NONE ok, we do not need to have the number of nodes. So, it will be NONE rather NULL then again a NULL. And finally, we must pass the address of the DM object. So, that is the final thing that you need to pass right.

(Refer Slide Time: 10:41)



```
1 Creates an object that will manage the communication of two-dimensional regular array data that is distributed across some processors.
2
3 Synopsis
4
5 #include "petscdmda.h"
6 PetscErrorCode DMCreate2d(MPI_Comm comm, DMBoundaryType bx, DMBoundaryType by, DMStencilType stencil_type,
7   PetscInt M, PetscInt N, PetscInt m, PetscInt n, PetscInt dof, PetscInt s, const PetscInt lx[], const PetscInt ly[], DM *da)
8 Collective
9
10 Input Parameters
11
12 comm - MPI communicator
13 bx,by - type of ghost nodes the array have. Use one of DM\_BOUNDARY\_NONE, DM\_BOUNDARY\_GHOSTED, DM\_BOUNDARY\_PERIODIC.
14 stencil_type- stencil type. Use either DMDA\_STENCIL\_BOX or DMDA\_STENCIL\_STAR.
15 M,N - global dimension in each direction of the array
16 m,n - corresponding number of processors in each dimension (or PETSC\_DECIDE to have calculated)
17 dof - number of degrees of freedom per node
18 s - stencil width
19 lx, ly - arrays containing the number of nodes in each cell along the x and y coordinates, or NULL. If non-null, these must be of length as m and n,
   and the corresponding m and n cannot be PETSC\_DECIDE. The sum of the lx[] entries must be M, and the sum of the ly[] entries must be N.
20
21 Output Parameter
22
23 da- the resulting distributed array object
24
25 Options Database Key
```

So, this is how we create a grid, so that is it pretty much for creating the object and this is what you need to do. Now, let us go ahead and create the matrix. So, we can no rather not the matrix, but let us proceed with the da first; so, DMSetFromOptions da. And this is important if we want to pass command line arguments on how large or how small we want the stencil to be.

If we do not care, I mean you have to do this; otherwise you cannot pass any command line arguments then DMSetup da. It is the same as the routine for setting up a matrix or a vector. Then DMCreateMatrix. Now, this step is quite important because it will tell the function that this structured grid has to eventually pass its arguments to A.

Note that we do not need to worry about the size of A, because depending on what this grid is going to be, the size of A will be automatically fixed. And why is that?

(Refer Slide Time: 12:17)

```
1 #include <petsc.h>
2
3
4 int main(int argc, char **argv)
5 {
6     DM da;
7     Mat A;
8     Vec b, u;
9     KSP ksp;
10    double err;
11    DMALocalInfo info; // Data structure to hold information about the array
12
13    PetscInitialize(&argc, &argv, NULL, "Solve Poisson");
14
15    DMDCreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, 1, 1, NULL, NULL, &da);
16
17    DMSetFromOptions(da);
18    DMSetUp(da);
19    DMCreateMatrix(da, A);
20
21    return PetscFinalize();
22 }
```

Star stencil

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta x^2}$$

DMDA → structured grid
distributed member data array

$$\vec{A} \vec{u} = \vec{b} \quad \begin{matrix} m \times n \\ m^1 \times m^2 \end{matrix}$$

matrix $(m \times n) \times (m \times n)$
 $m^1 \times m^2$

Because, if there are m cross; if it is a grid of $m \times n$ size in the matrix, size will be $m \times n \times m \times n$. If m and n are equal, so if it is a 9 by 9 grid, then the number of variables will be actually 81. So, it is m square by n square in this particular case. So, A will automatically be sort of linked to the structured grid. So, we do not need to worry about that anymore and finally, the `MatSetFromOptions A`. So, this is where you sort of create the matrix A alright.

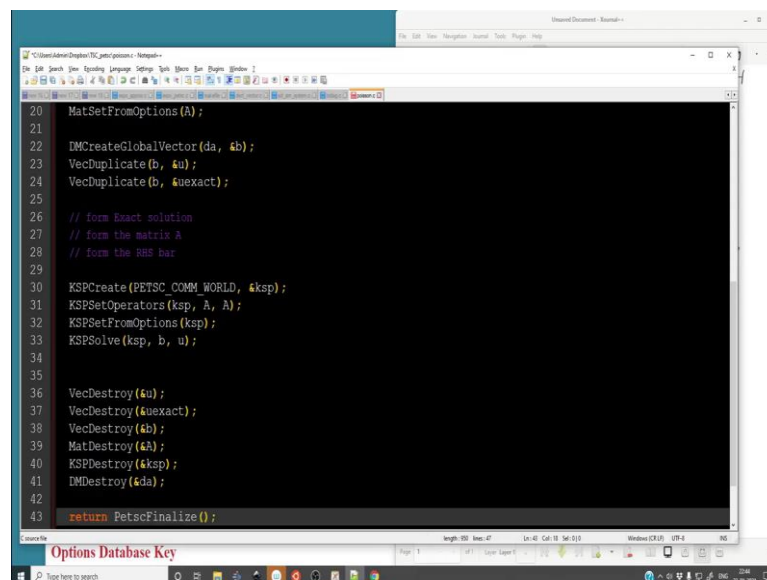
(Refer Slide Time: 13:01)

```
4 int main(int argc, char **argv)
5 {
6     DM da;
7     Mat A;
8     Vec b, u, uexact;
9     KSP ksp;
10    double err;
11    DMALocalInfo info; // Data structure to hold information about the array
12
13    PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
14
15    DMDCreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, DMDC_STENCIL_STAR, 9, 9,
16    PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL, &da);
17
18    DMSetFromOptions(da);
19    DMSetUp(da);
20    DMCreateMatrix(da, A);
21    MatSetFromOptions(A);
22
23    DMCreateGlobalVector(da, &b);
24    VecDuplicate(b, &u);
25
26    return PetscFinalize();
27 }
```

So, once we have this, we can now create the vectors. So, what we can do is `DMCreateGlobalVector da`, and will pass the address of `b` then we will do a `VecDuplicate da` comma the address not `da` it has to be `b` comma address of `u`. So, this is just to initialize `u`, and because eventually we want to compare the solutions, well, let us make it `uexact`. So, `uexact` will hold the exact solution. So, let us create another duplicate `b` comma address of `uexact`.

So, so far we have created `A`, `b`, `u`, and `uexact`. Based on this, we have to proceed. So, now, at this point, we have sort of linked `A` to the `da`, we have linked `b` to `da`, we have linked `u` to `da`, and we have linked `uexact` to `da`.

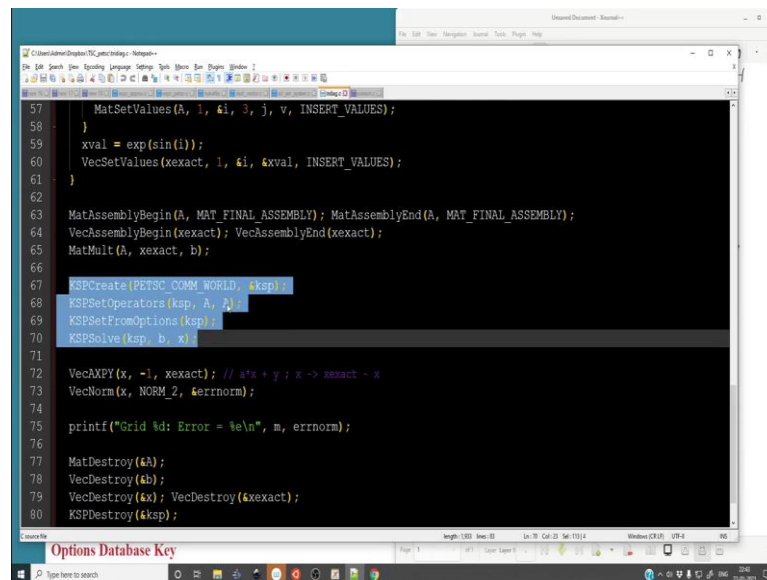
(Refer Slide Time: 14:21)



```
20 MatSetFromOptions(A);
21
22 DMCreateGlobalVector(da, &b);
23 VecDuplicate(b, &u);
24 VecDuplicate(b, &uexact);
25
26 // form exact solution
27 // form the matrix A
28 // form the RHS bar
29
30 KSPCreate(PETSC_COMM_WORLD, &ksp);
31 KSPSetOperators(ksp, A, A);
32 KSPSetFromOptions(ksp);
33 KSPSolve(ksp, b, u);
34
35
36 VecDestroy(&u);
37 VecDestroy(&uexact);
38 VecDestroy(&b);
39 MatDestroy(&A);
40 KSPDestroy(&ksp);
41 DMDestroy(&da);
42
43 return PetscFinalize();
```

What needs to be done is we need to do multiple things now. We need to form the exact solution for the future reference, then we need to form the matrix `A`, we need to form the RHS matrix vector `b`, and finally, we need to solve it. So, how do we solve it? Again it is the same as before. And in this case, we can reuse some of the code.

(Refer Slide Time: 14:51)

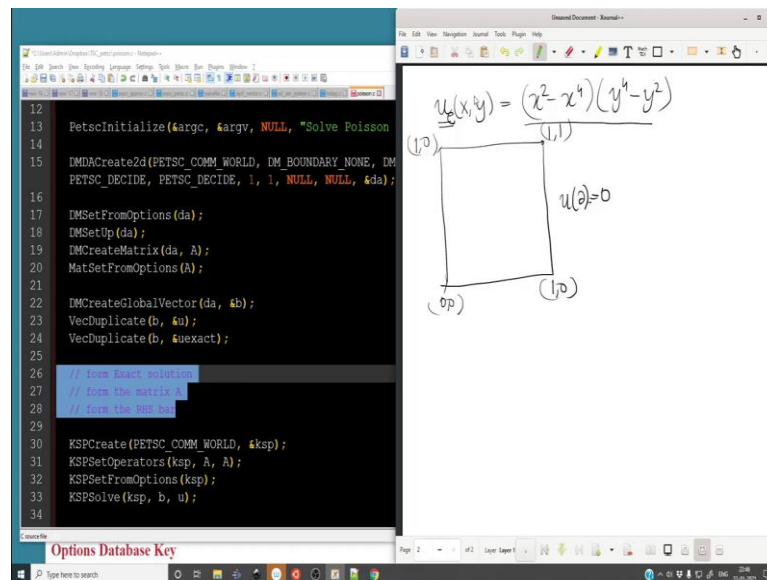


```
57 MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
58 }
59 xval = exp(sin(i));
60 VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
61 }
62
63 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
64 VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
65 MatMult(A, xexact, b);
66
67 KSPCreate(PETSC_COMM_WORLD, &ksp);
68 KSPSetOperators(ksp, A, A);
69 KSPSetFromOptions(ksp);
70 KSPSolve(ksp, b, x);
71
72 VecAXPY(x, -1, xexact); // a*x + y; x -> xexact - x
73 VecNorm(x, NORM_2, &errnorm);
74
75 printf("Grid %d: Error = %e\n", m, errnorm);
76
77 MatDestroy(&A);
78 VecDestroy(&b);
79 VecDestroy(&x); VecDestroy(&xexact);
80 KSPDestroy(&ksp);
```

So, this particular code will be useful. So, we will simply have KSPCreate PETSC COMM WORLD pass the address of the ksp object. Then KSPSetOperators ksp, A, A the preconditional is going to be A default SetFromOptions ksp, then ksp b comma instead of x we will now have u alright. So, this is the entire thing. But now we need to focus on writing these things.

But before even doing that let us quickly free the vectors before the code finishes. So, VecDestroy pass the address of u, copy this, uexact b MatDestroy A. Now, what else do we have? Not A, but the address of A. Then we have KSPDestroy address of ksp. Finally, we have additional DMDDestroy address of da, and then finally, we have return PetscFinalize, so that is as usual. So, now we must go ahead and write down these functions alright. So, like we had done in the case of the code that we did in Python.

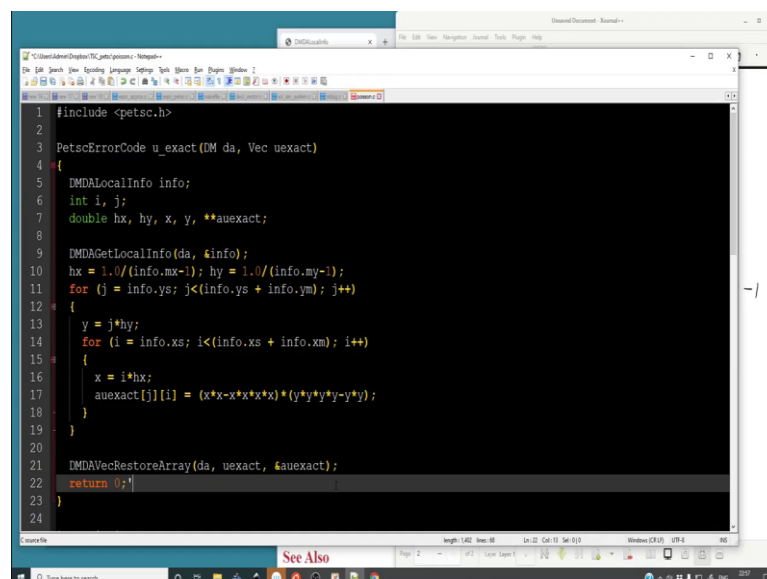
(Refer Slide Time: 16:35)



We going to assume a solution u . x, y is the exact same solution, you can compare it later on. So, it is $(x^2 - x^4)(y^4 - y^2)$. It is a function which satisfies the Dirichlet boundary conditions on a unit square. This is $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(0, 1)$. So, it satisfies the Dirichlet boundary condition u on the boundaries. So, on the boundaries is equal to 0 alright.

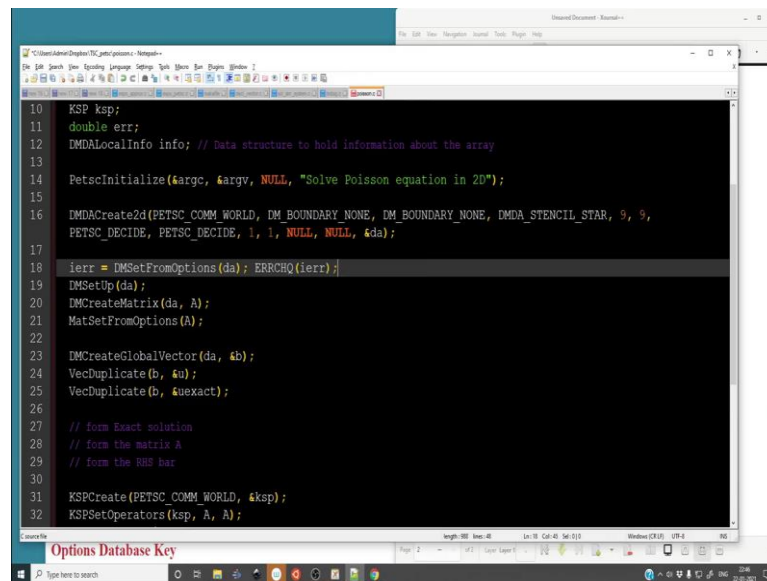
So, what we need to do is encode this into the `uexact` vector, so that we can later on compare it right. So, let us create that function. So, let us create that function.

(Refer Slide Time: 17:39)



So, let us make the return value PetscErrorCode, and we have used this i error actually.

(Refer Slide Time: 17:51)

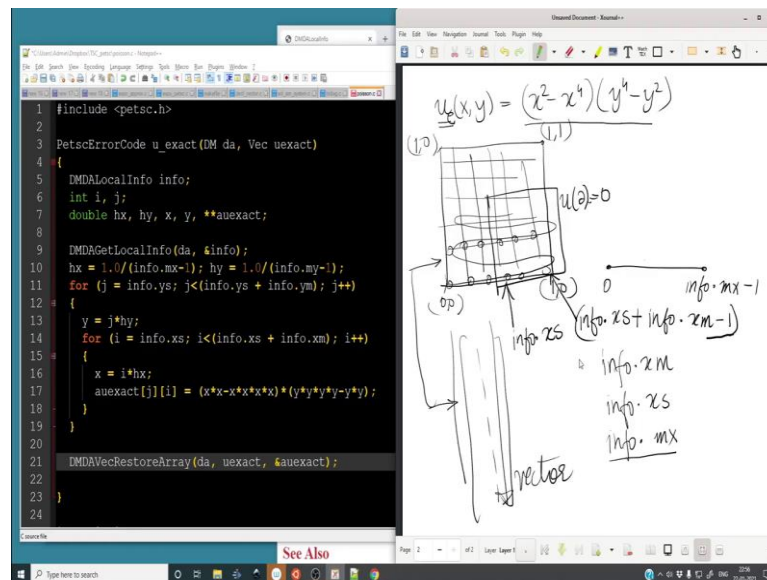


```
10 KSP ksp;
11 double err;
12 DMDataLocalInfo info; // Data structure to hold information about the array
13
14 PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
15
16 DMDataCreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE, DMData_STENCIL_STAR, 9, 9,
17 PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL, &da);
18
19 ierr = DMDataSetFromOptions(da); ERRCHQ(ierr);
20 DMDataSetUp(da);
21 DMDataCreateMatrix(da, A);
22 MatSetFromOptions(A);
23
24 DMDataCreateGlobalVector(da, &b);
25 VecDuplicate(&b, &u);
26 VecDuplicate(&b, &uexact);
27
28 // form exact solution
29 // form the matrix A
30 // form the RHS bar
31
32 KSPCreate(PETSC_COMM_WORLD, &ksp);
33 KSPSetOperators(ksp, A, A);
```

I have not been using it over here. Each function call can be something like this, and then we can do a ERRCHQ of ierr. But I am not doing it, you do not need to do it. But in case you need to make a large program, you need to debug, you need to do all that, and then you have to declare the PetscErrorCode ierr over here. And we have done it in the very beginning, but I am not going to do it over here.

So, let us create a function called as CREATE or u exact, and it will take as an input the structured grid da and the vector uexact. So, uexact has not been initialized. So, the purpose of this particular function that is uexact is to initialize it, and have it available in the main as well. So, we are passing the vector, we are going to modify the vector. So, essentially this is the input, and this both the input and the output. The input is coming as nothing but the output will be the exact value.

(Refer Slide Time: 19:07)



Now, we have to loop over the grid, because look whenever we want to calculate this, we need to know how the loop will go on, we need to know how many grids have been initialized. Well, we have created 9 by 9, but you can pass command and arguments and change it.

So, over here we are going to create another local copy of DMDALocalInfo. If we going to call it as Info, so note that this particular local copy does not conflict with this because of the scope of this function being different from the scope of the main function. Then we are going to define int i, j for running the double loop, and we are going to create hx, hy because hx and hy are simply the grid spacing.

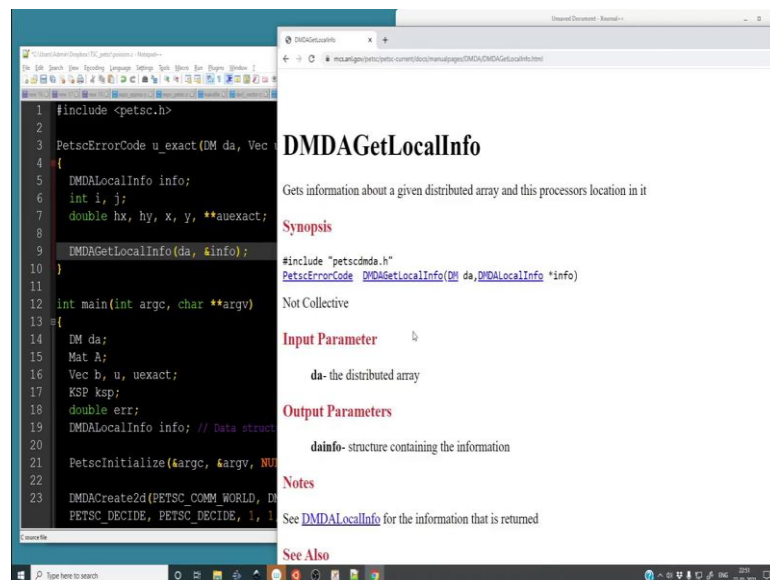
We are going to pass, we are going to create x and y. And we are going to create a double pointer which will act as an auxiliary a matrix to store uexact. So, basically what we are going to do is see uexact is a vector. So, in the PETSc data structure, it is going to be a vector. But in reality uexact is a matrix it has values all over the grid.

So, what we can do is, we can swap between this 2D representation which is in the form of a double pointer, because arrays or rather matrices in C or in the form of double pointers. And we can convert back and forth between the vec structure of PETSc and the matrix structure that we require in C. So, for that, we need to create a double array; and the double array we will call it auexact. So, a stands for the auxiliary matrix for that, alright.

So, `DMDAGetLocalInfo` is the function which picks out the information from the structured grid `da` and passes it on to the variable `info`. So, now, what does `info` have? `Info` has two things; one is `info dot x m`, then it has `info dot xs`, it has `info dot m x`. So, `info dot m x` is the maximum. So, if the grid has `m x` number of points, so the grid will go from 0 to `info dot m x - 1` that is what it means alright.

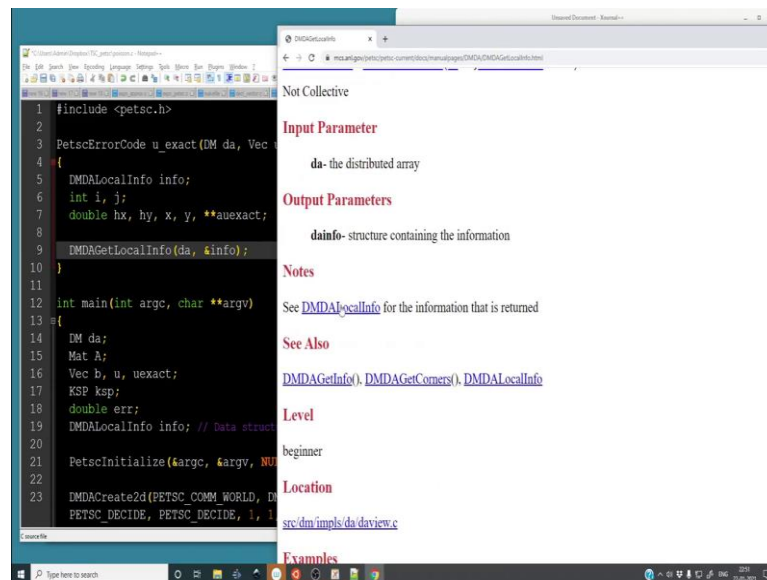
So, we must run a loop going from `info dot y s`. So, what is rather not `y s`, `xs`, `xs` is like the starting point. So, if one processor is dealing with only this thing. So, for this the local starting index for that processor will be `info dot xs` ok. It is like a local starting index. And this particular last index will be `info dot xs + info dot xm - 1`, this will be the ending index.

(Refer Slide Time: 22:59)



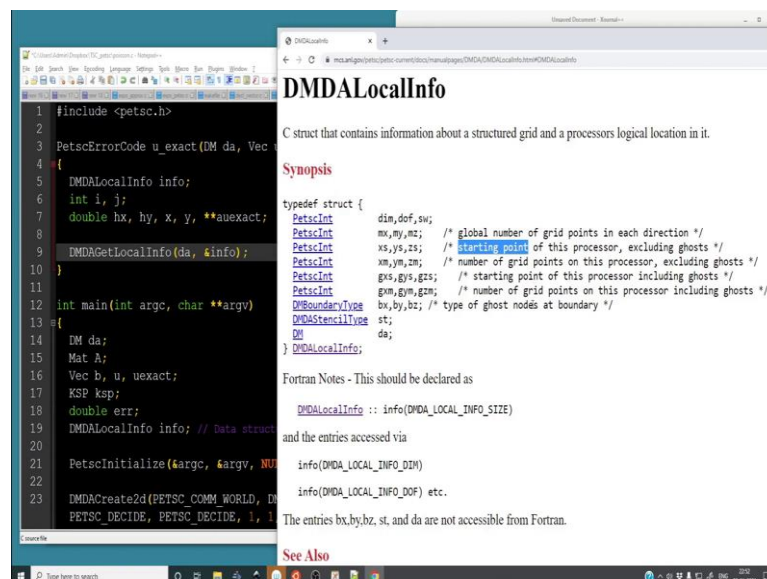
So, let me just go to the function reference. So, what is it? `DMDAGetLocalInfo`.

(Refer Slide Time: 23:09)



So, it contains all these things.

(Refer Slide Time: 23:13)



So, it is the global number of grid points. So, it is a structure, it is a data structure which contains all these informations. So, the dimension, the degree of freedoms alright, then we have the globals number of grid points. For that processor, what is the starting point? For that processor, the number of grid points, if you have ghost nodes ok, type of ghost nodes, the stencil, da, and all this thing.

So, for our purpose, we can make do with the starting point of that particular processor to the end point on that processor rather than the number of. So, that is why we have plus. So, number of grid points plus starting point will be the final point, and -1 will be the index of that point that is why this -1 because it is the index of the ending point on that processor. If you are using one processor, then it is the maximum value -1. So, just think about it how it will work.

So, before that let us create $hx = 1.0 / \text{info dot } mx - 1$. So, look $\text{info dot } mx$ is the maximum number of grid points. So, these grid spacing will be this hy will be 1.0 divided by $\text{info dot } my - 1$. So, these are the two hx and hy . Accordingly we can now define x as $i \times hx$ and y as $j \times hy$ ok, the same thing we had done in python as well alright.

So, let us do a loop for $j = \text{info dot } ys$ $j < \text{info dot } ys + \text{info dot } ym$, let us just create a bracket just to be super sure and $j++$. Let us now create another for loop for i equal to. So, let me just copy this. We just need to replace everything by the x . And so this will be xs , and this will be xs , this will be i , this will be xm , and this will be $i++$. So, there is a huge chance you can mess this up because if you forget to change one i or one j , all the loops will run incorrectly.

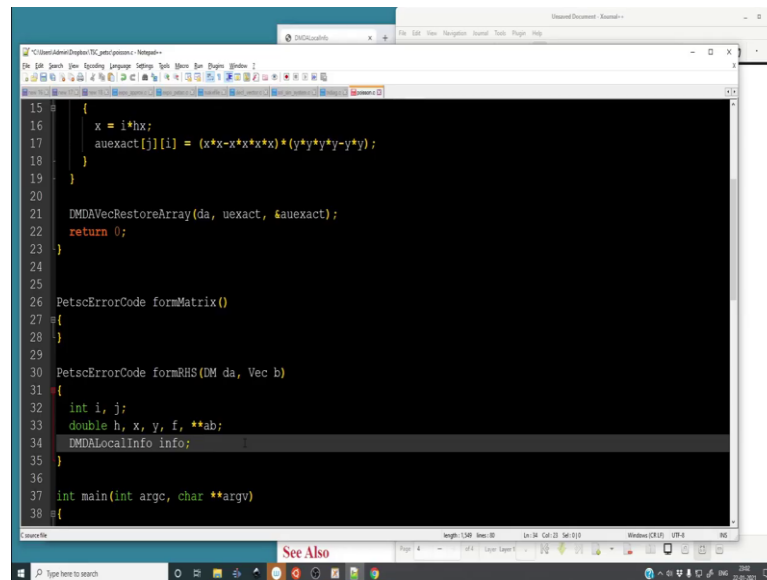
So, before entering into this, let us just define $y = j \times hy$. Before going into this loop, let us define x as $i \times hx$ great. So, then what we can do is we can say $u_{exact}(j, i)$. So, the j th row i th column is equal to the exact solution which is $x^2 - y^2$ times $x^2 - x^4$. So, this is it times $y^2 - y^4$ alright.

I mean you could use power and all that, but it is simple example we can get it done like this as well. So, that is the analytical solution what we have written over here alright. So, now, that we have defined the u_{exact} as a two-dimensional grid ok; look this variable is actually a two-dimensional grid right now.

So, what we have to do is assign that grid to the vector u_{exact} . And the way to do it is `DMDAVecRestoreArray` over da , u_{exact} , and the address of u_{exact} . This particular line right, this particular line will tell the function that look u_{exact} is a vector based on the structured grid da . And this is the actual matrix u_{exact} which is based on the structured grid da .

So, it will do the interconversion for you. Essentially it will flatten out the solution in the form of a single vector. It will take all these entries and flatten them out into a single vector that is what it will do. And finally, if everything is successful, let us simply return 0 alright good.

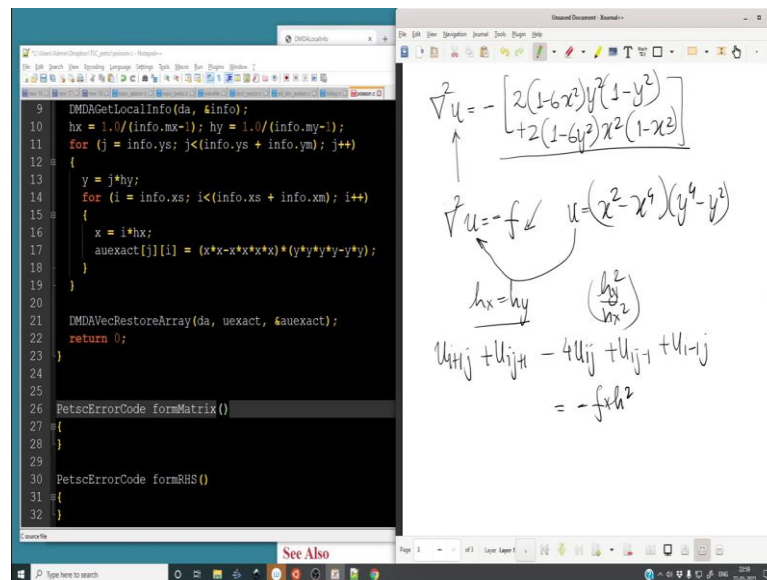
(Refer Slide Time: 28:33)



```
15 {
16     x = i*hx;
17     auexact[j][i] = (x*x-x*x*x*x)*(y*y+y*y-y*y);
18 }
19 }
20
21 IMDAVecRestoreArray(da, uexact, &auexact);
22 return 0;
23 }
24
25
26 PetscErrorCode formMatrix()
27 {
28 }
29
30 PetscErrorCode formRHS(DM da, Vec b)
31 {
32     int i, j;
33     double h, x, y, f, **ab;
34     IMDALocalInfo info;
35 }
36
37 int main(int argc, char **argv)
38 {
```

So, that takes care of PetscErrorCode uexact function right. Now, let us create a function which will create the right hand side. So, PetscErrorCode form RHS. And we will also need a function PetscErrorCode formMatrix. These are the two functions we still need to write And so what should these things contain? So, let us first look at I mean we have done this for the case of python already, but still let us have some.

(Refer Slide Time: 29:25)



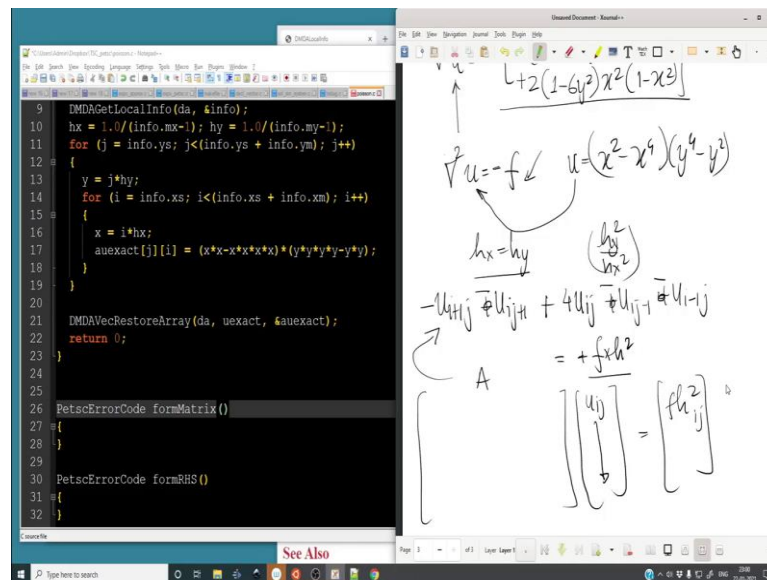
For completeness final equation look something like this $-[2(1-6x^2)y^2(1-y^2) + 2(1-6y^2)x^2(1-x^2)]$. So, how did this thing come into being? Well, we made use of the fact that we know the exact solution u ok. So, the equation is this equal to $-f$.

And we know that the solution u is $(x^2 - x^4)(y^4 - y^2)$. We know that this is the solution. So, we can plug this into this, and find out the function f which satisfies it. So, this is the function that satisfies it alright.

So, on the left hand side, you have $u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i,j-1} + u_{i-1,j} = -f \times h^2$. Well, let us assume that h_x and h_y will be equal for this particular specific case. We do not want to make it too general; otherwise, it will create nothing.

We just have a ratio of h_y/h_x square appearing somewhere over in these terms. But anyway for now we can go ahead with this assumption, and it is fair to have a structured grid with equal points in both the directions.

(Refer Slide Time: 31:19)

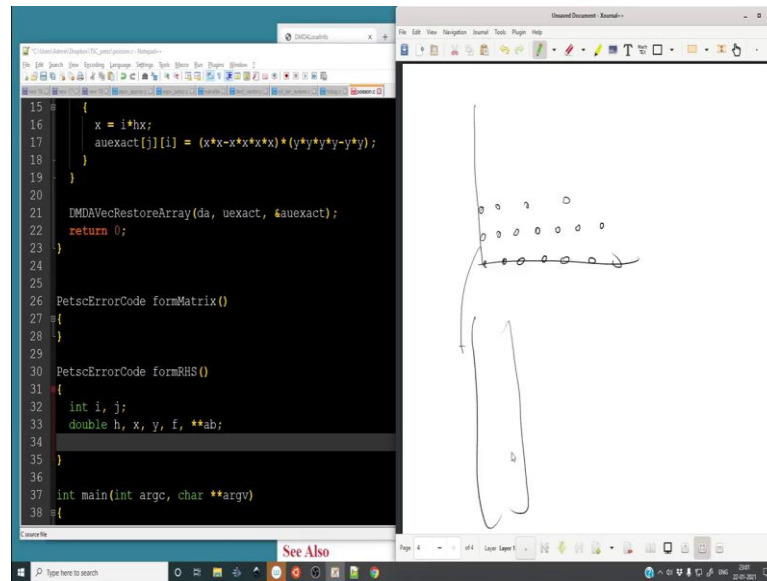


So, what is the right hand side? So, the right hand side is simply going to be h^2 times this function f . Well, what we can do is take this minus sign over here. So, this becomes minus, this becomes minus, this becomes plus, minus and minus. So, we have this is equal to this right hand side.

So, this vector A will be constructed as per this left hand side, and so because it is going to be u_{ij} all the elements u_{ij} s right hand side will be simply f times h square at the point ij . So, we can first go ahead and create the function form RHS. So, again what are we going to require.

So, we are going to require two iterators i and j , then double h comma x comma y comma, we can create f and a matrix to hold the value of the right hand side over all the grid points. So, once again eventually what you are going to do is convert this matrix because once again I am going to reiterate this, and open intended.

(Refer Slide Time: 32:29)

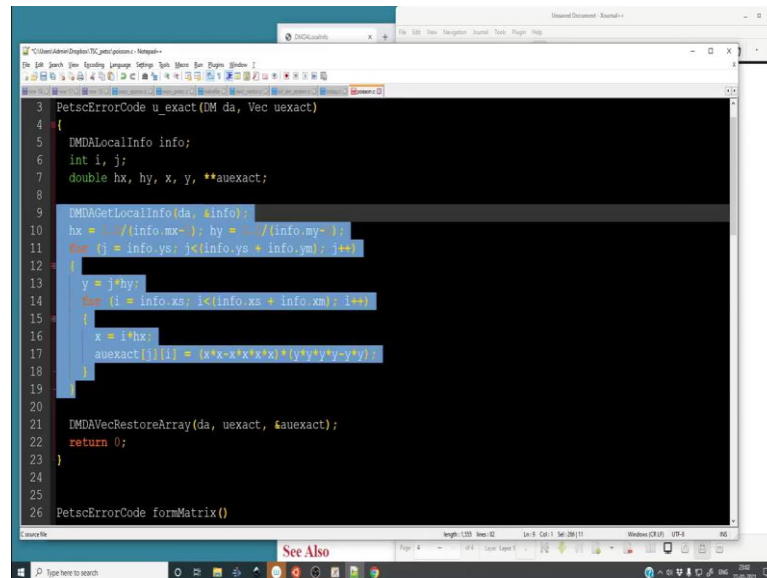


So, I am going to we are going to have the right hand side at all the grid points alright. We are going to essentially have right hand side and all the grid points, and we have to convert this into a vector structure which is there in PETSc. So, the inputs will be the DM da, and the vector b that is going to be I mean these are not the inputs, but these are just placeholders for the input.

When we call the function from main, we are going to call it as form RHS da comma b. This is just the declaration of the function and the definition inside alright. So, these are things I mean quite usual to C programs alright. So, now we have to create this RHS function. So, let us do this. So, double we have defined.

We will need the DMDALocalInfo, we will call it info again. This scope is different from the scope of the previous function, and main both alright. Then we actually need this double loop.

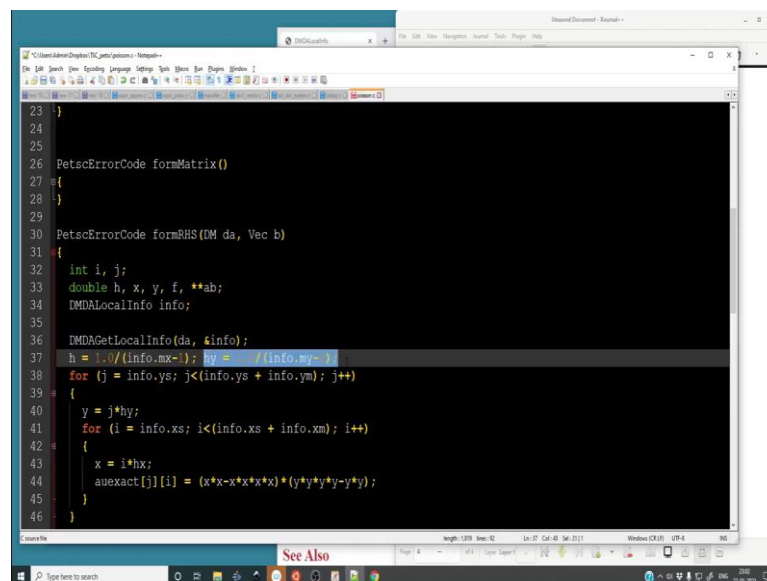
(Refer Slide Time: 33:43)



```
3 PetscErrorCode u_exact(DM da, Vec uexact)
4 {
5     DMDataLocalInfo info;
6     int i, j;
7     double hx, hy, x, y, **auexact;
8
9     DMDataGetLocalInfo(da, &info);
10    hx = 1.0/(info.mx-1); hy = 1.0/(info.my-1);
11    for (j = info.ys; j < (info.ys + info.ym); j++)
12    {
13        y = j*hy;
14        for (i = info.xs; i < (info.xs + info.xm); i++)
15        {
16            x = i*hx;
17            auexact[i][j] = (x*x-x*x*x*x)*(y*y+y-y*y);
18        }
19    }
20
21    DMDataVecRestoreArray(da, uexact, &auexact);
22    return 0;
23 }
24
25
26 PetscErrorCode formMatrix()
```

So, let us paste this double loop. We are going to need it anyway.

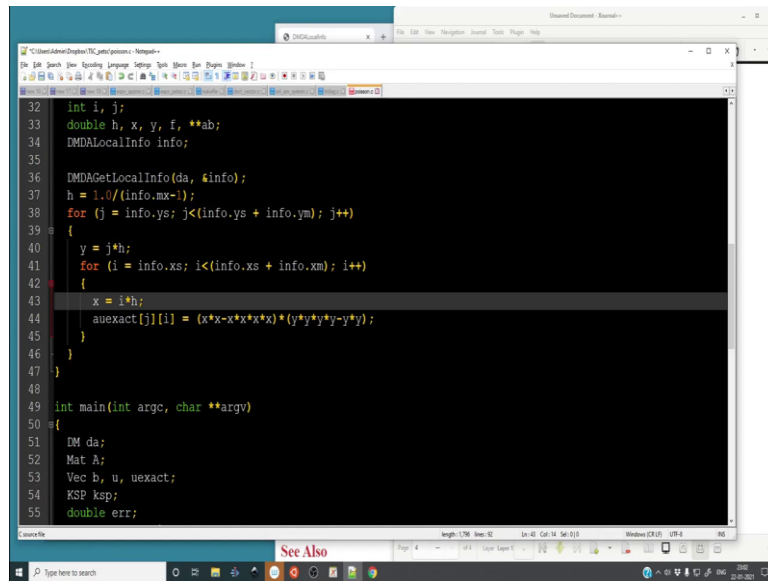
(Refer Slide Time: 33:47)



```
23 }
24
25
26 PetscErrorCode formMatrix()
27 {
28 }
29
30 PetscErrorCode formRHS(DM da, Vec b)
31 {
32     int i, j;
33     double h, x, y, f, **ab;
34     DMDataLocalInfo info;
35
36     DMDataGetLocalInfo(da, &info);
37     h = 1.0/(info.mx-1); hy = 1.0/(info.my-1);
38     for (j = info.ys; j < (info.ys + info.ym); j++)
39     {
40         y = j*hy;
41         for (i = info.xs; i < (info.xs + info.xm); i++)
42         {
43             x = i*hx;
44             auexact[j][i] = (x*x-x*x*x*x)*(y*y+y-y*y);
45         }
46     }
47 }
```

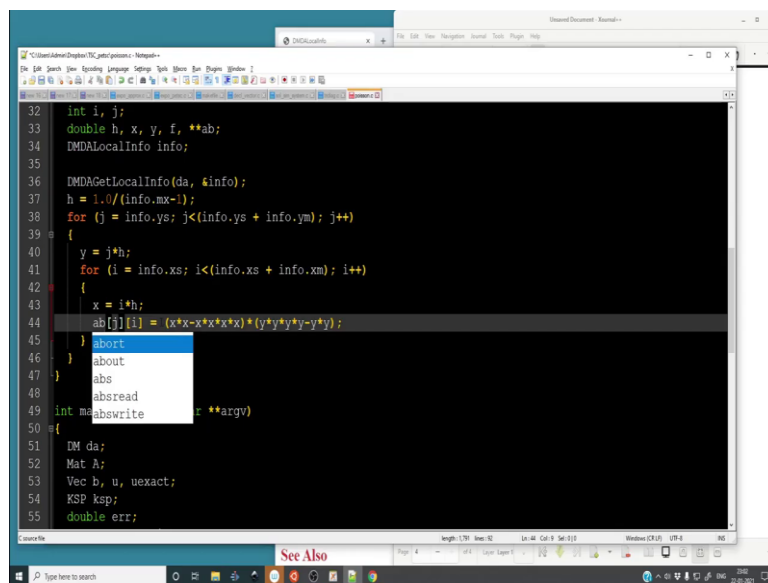
So, it is going to do this. Instead of having these two declarations, we are going to just keep one declaration. The j loop remains the same; instead of hy and hx , we simply have this and ab , i .

(Refer Slide Time: 34:03)



```
32 int i, j;
33 double h, x, y, f, **ab;
34 DMDALocalInfo info;
35
36 DMDAGetLocalInfo(da, &info);
37 h = 1.0/(info.mx-1);
38 for (j = info.ys; j<(info.ys + info.ym); j++)
39 {
40     y = j*h;
41     for (i = info.xs; i<(info.xs + info.xm); i++)
42     {
43         x = i*h;
44         auxact[j][i] = (x*x-x*x*x*x)*(y*y*y*y-y*y);
45     }
46 }
47
48
49 int main(int argc, char **argv)
50 {
51     DM da;
52     Mat A;
53     Vec b, u, uexact;
54     KSP ksp;
55     double err;
```

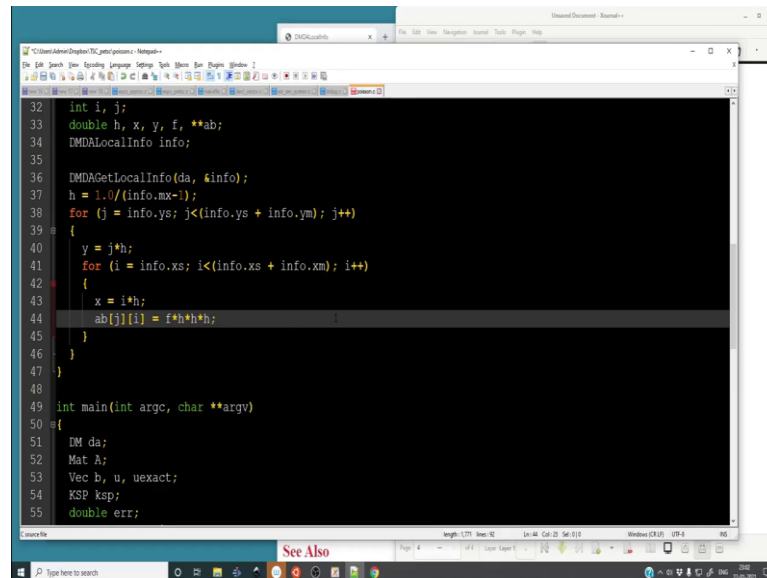
(Refer Slide Time: 34:07)



```
32 int i, j;
33 double h, x, y, f, **ab;
34 DMDALocalInfo info;
35
36 DMDAGetLocalInfo(da, &info);
37 h = 1.0/(info.mx-1);
38 for (j = info.ys; j<(info.ys + info.ym); j++)
39 {
40     y = j*h;
41     for (i = info.xs; i<(info.xs + info.xm); i++)
42     {
43         x = i*h;
44         ab[j][i] = (x*x-x*x*x*x)*(y*y*y*y-y*y);
45     }
46 }
47 }
48 }
49 int main(int argc, char **argv)
50 {
51     DM da;
52     Mat A;
53     Vec b, u, uexact;
54     KSP ksp;
55     double err;
```

Will be equal to then f times h square.

(Refer Slide Time: 34:11)

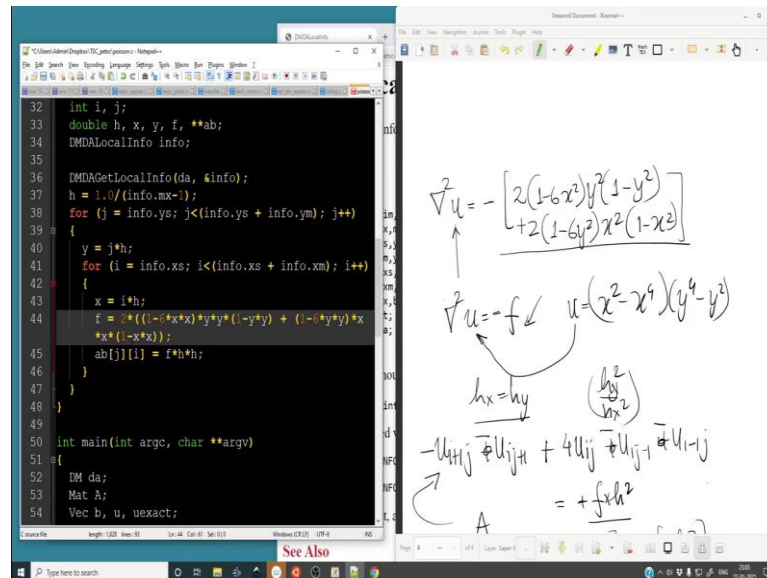


```
32 int i, j;
33 double h, x, y, f, **ab;
34 DMDALocalInfo info;
35
36 DMDAGetLocalInfo(da, &info);
37 h = 1.0/(info.mx-1);
38 for (j = info.ys; j<(info.ys + info.yh); j++)
39 {
40     y = j*h;
41     for (i = info.xs; i<(info.xs + info.xh); i++)
42     {
43         x = i*h;
44         ab[j][i] = f+h*h*h;
45     }
46 }
47
48
49 int main(int argc, char **argv)
50 {
51     DM da;
52     Mat A;
53     Vec b, u, uexact;
54     KSP ksp;
55     double err;
```

Where f will be equal to two times, now what is that pick down this whole term. So, what is it? Let me reduce this in size $2 \times 1 - 6 \times x^2$, and make it into a bracket $\times y \times y \times 1 - y \times y$. Let me copy this because the other term is simply x and y swapped, this will be plus this. So, what do we have $1 - y \times y \times x \times x \times x$ alright. So, this is what f is. And the right hand side is $f \times h$ square alright great.

So, lastly what should? We do we should restore this double matrix to the vector b . So, we must do `DMDAVecRestoreArray`, and we must pass the da , b , and the address of the double matrix, great. Lastly, we must return the 0. So, that takes care of the right hand side function as well. And now it is simply a question of creating the matrix that is quite simple. Given we have done the creation of the vectors.

(Refer Slide Time: 35:55)



This should be not that difficult. So, DMDALocalInfo, info, again we are going to need int i, j as the iterators double h. Now, we have to insert 5 elements 1, 2, 3, 4, 5. So, each row there will be 5 insertions into the matrix. So, you must create a value matrix which is of size 5, yeah. And apart from this, because we are going to fill in the matrix there is an additional data structure called as MatStencil. We will create it as row and col we will have 5.

So, it will sort of link the da and the matrix A. So, we have to tell we have to pass this MatStencil structure into the mat set values. So, once I write the code it, I will explain it how it works. Before that, I can just show you the quick functional reference or rather the data structure reference.

(Refer Slide Time: 37:17)

```

20
21 DMDAvecRestoreArray(da, uexact, &uexact);
22 return 0;
23 }
24
25
26 PetscErrorCode formMatrix()
27 {
28 DMDALocalInfo info;
29 int i, j;
30 double h, v[5];
31 MatStencil row, col[5];
32
33 DMDAGetLocalInfo(da, &info);
34
35 }
36
37 PetscErrorCode formRHS(DM da, Vec b)
38 {
39 int i, j;
40 double h, x, y, f, **ab;
41 DMDALocalInfo info;
42
43 DMDAGetLocalInfo(da, &info);

```

So, it is a data structure for storing information about a single row or single column of a matrix.

(Refer Slide Time: 37:23)

$\nabla u = -f$
 $u = (x^2 - x^4)(y^4 - y^2)$
 $h_x = h_y$
 $A = \begin{bmatrix} -4 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = fh^2$

So, row i, j column i, j . So, those i, j s are from the actual grid, but row will calculate the linear index from i, j and column will also calculate the linear index from i, j . So, it is just allowing you to seamlessly do those linear indexing business without having to write any extra code.

Essentially, you are still looking over the you are still doing a loop over nothing but the grid, but you are able to pass the appropriate index to the through the matrix A alright. So, that is quite beneficial alright. So, we have created this. Now, what else do we need ok? We will declare everything that we need as we go along.

So, DMDAGetLocalInfo pass the information from da to the address of info. Again this info is local. So, essentially we have done this line.

(Refer Slide Time: 38:31)

```

26 PetscErrorCode formMatrix()
27 {
28     DMDALocalInfo info;
29     int i, j;
30     double h, v[5];
31     MatStencil row, col[5];
32
33     DMDAGetLocalInfo(da, &info);
34     h = 1.0/(info.mx-1);
35 }
36
37 PetscErrorCode formRHS(DM da, Vec b)
38 {
39     int i, j;
40     double h, x, y, f, **ab;
41     DMDALocalInfo info;
42
43     DMDAGetLocalInfo(da, &info);
44     h = 1.0/(info.mx-1);
45     for (j = info.ys; j<(info.ys + info.yh); j++)
46     {
47         y = j*h;
48         for (i = info.xs; i<(info.xs + info.xh); i++)
49     {

```

We also need this particular line alright.

(Refer Slide Time: 38:37)

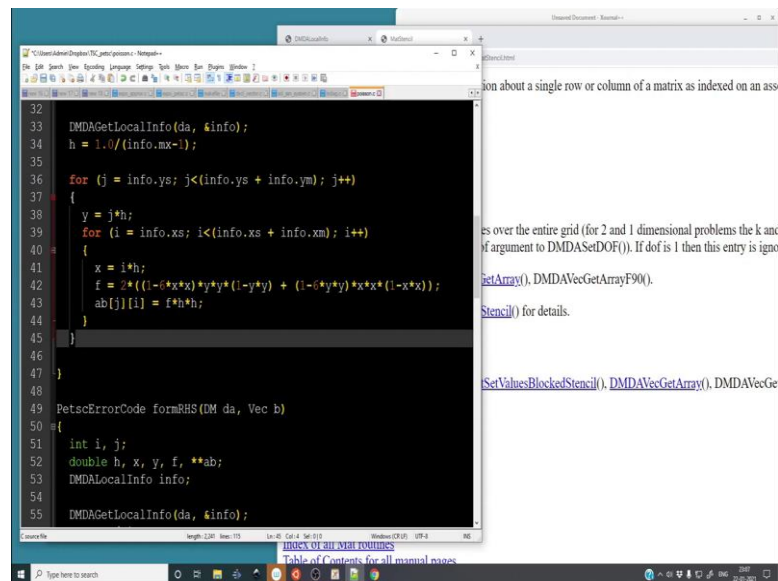
```

45     for (j = info.ys; j<(info.ys + info.yh); j++)
46     {
47         y = j*h;
48         for (i = info.xs; i<(info.xs + info.xh); i++)
49         {
50             x = i*h;
51             f = 0.5*(1.0-x*x)*y*y*(1.0-y*y) + (1.0-y*y)*x*x*x*(1.0-x*x);
52             ab[j][i] = f*h*h;
53         }
54     }
55 }

```

Then we are going to actually need this particular double loop as well alright.

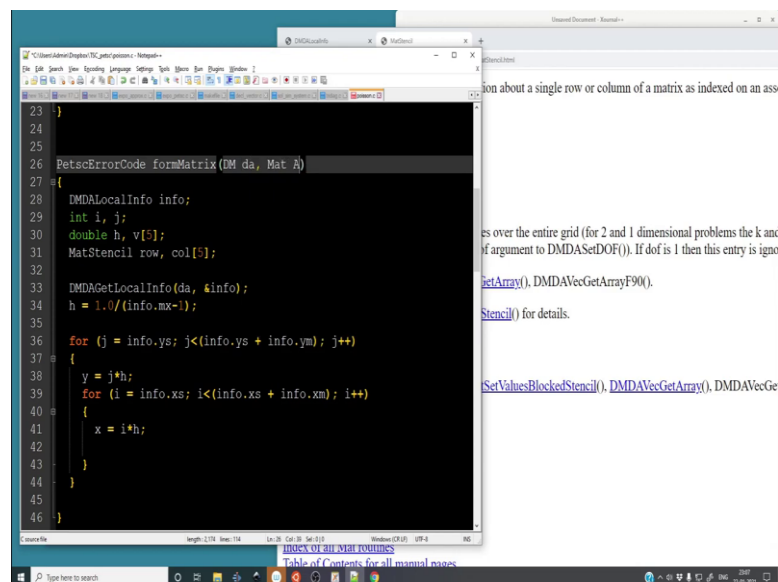
(Refer Slide Time: 38:43)



```
32
33 DMDAGetLocalInfo(da, &info);
34 h = 1.0/(info.mx-1);
35
36 for (j = info.ys; j<(info.ys + info.ym); j++)
37 {
38     y = j*h;
39     for (i = info.xs; i<(info.xs + info.xm); i++)
40     {
41         x = i*h;
42         f = 2*((1-phi*x*x)*y*y*(1-y*y) + (1-phi*y*y)*x*x*(1-x*x));
43         ab[j][i] = f*h*h;
44     }
45 }
46
47 }
48
49 PetscErrorCode formRHS(DM da, Vec b)
50 {
51     int i, j;
52     double h, x, y, f, **ab;
53     DMDALocalInfo info;
54     DMDAGetLocalInfo(da, &info);
55
```

But now in the double loop, we have to do something else. So, this goes alright. So, now, what should be the inputs to the function?

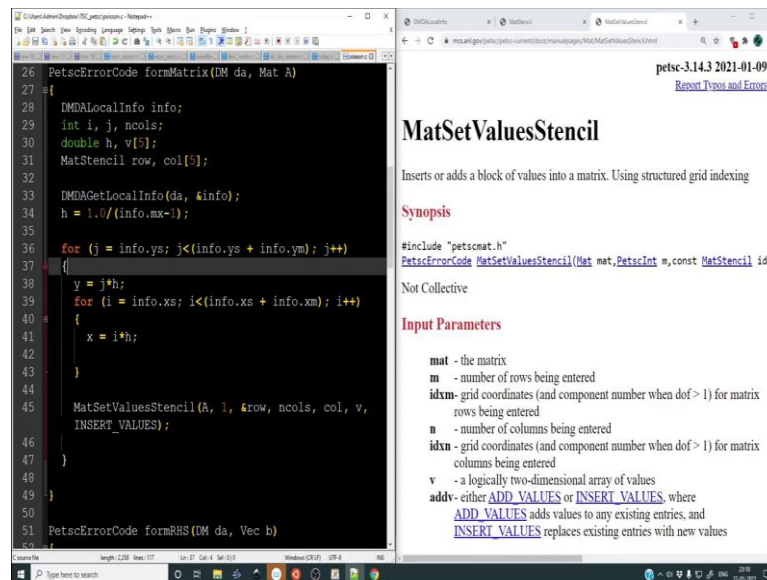
(Refer Slide Time: 38:53)



```
23 }
24
25
26 PetscErrorCode formMatrix(DM da, Mat A)
27 {
28     DMDALocalInfo info;
29     int i, j;
30     double h, v[5];
31     MatStencil row, col[5];
32
33     DMDAGetLocalInfo(da, &info);
34     h = 1.0/(info.mx-1);
35
36     for (j = info.ys; j<(info.ys + info.ym); j++)
37     {
38         y = j*h;
39         for (i = info.xs; i<(info.xs + info.xm); i++)
40         {
41             x = i*h;
42
43         }
44     }
45 }
46
```

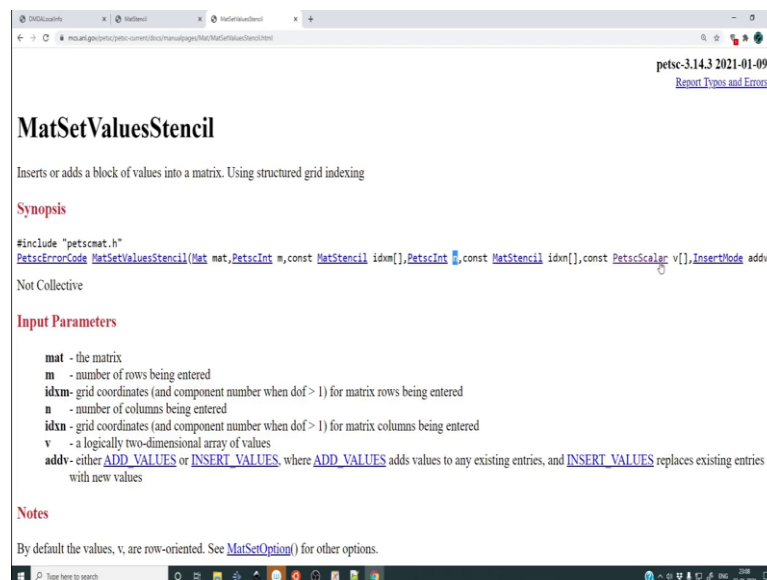
It has to be the DM da and matrix A alright. So, yeah, so, now we must do something over here. And at the end of the column loop, so this is the column loop, at the end of the column, we must set values.

(Refer Slide Time: 39:15)



So, `MatSetValuesStencil`. And this particular `MatSetValuesStencil` actually links the setting of the matrix A from the grid points, right.

(Refer Slide Time: 39:37)



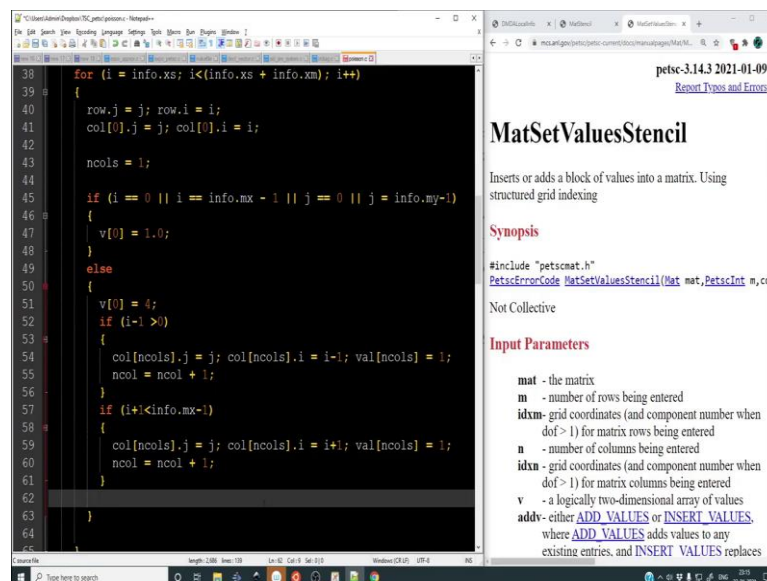
So, we must do this. So, first of all what does this contain `mat` set from on a `MatSetValuesStencil`. So, it, we must give the matrix the number of rows being entered. So, we are going to insert one row at a time. And what do we have ok. And what do we have, we have the id, the address server, the number of columns being entered. So, for

the interior points, it will be 5; for the boundary values, it will be only 1 alright, then PetscScalar and InsertMode. So, it is quite similar to whatever we have done great.

So, MatSetValuesStencil, this will contain A 1 row we will pass the address of row. Now, we have we must tell how many columns we are going to insert. So, let us create ncols as the number of columns you want to enter. So, then we must pass the address of the what the address col, because col is already a pointer, then the v array, and INSERT VALUES that is pretty much it.

Now, there must be a logic over here how to find out how many number of columns you have to fill. So, for the interior points, you have to only fill so you have to fill 5, but otherwise you need to fill 1 alright. So, we must declare ncols as an integer as well. So, ncols is an integer great. So, then what do we do? So, for this we do not need x and y anymore, you do not need it then ok.

(Refer Slide Time: 41:39)



So, row dot j is j; row dot i is going to be i, because we are focusing on this. So, column is actually a vector. So, column 0 dot j will be equal to j; column 0 dot i will be equal to i. So, this is just to tell that based on this row and column you figure out the linear index, do not make me do that ok. So, both these lines we will figure out the linear index alright.

So, now, we will start with $ncols = 1$. So, if nothing else happens, we will through this simply set the diagonal element. So, just a quick recap, what will be that diagonal element? If you once heard the linear index corresponding to this which is $i + n \times$ or rather $m \times j$, you are absolutely correct. So, it will put the value v_0 into that element if we leave the code at this juncture.

But what should we do? If $i = 0$ right or what i so this is the $x = 0$ point or if $i = info \cdot mx - 1$ or $j = 0$ or $j = info \cdot my - 1$. So, essentially these are all the boundaries. Now, what we have to do? If i is this, if j is this, then we must simply set v_0 to 1; we do not need to do anything else.

And simply this particular function when it executes, it will enter one in that particular location corresponding to that grid the boundary grid, and we are happy. But if this is not the case, alright, if this is not the case, then we have to do something else, else. We have to do something else.

Well. So, we must do $v_0 = 4$, then so that is it. So, now, if we are in the neighbourhood of a boundary, then again we must do something special. So, I request you to look at I mean work it through whatever I am going to write, you need to really work it through. So, this is for an interior point.

But if you have to deal with boundaries, you must check whether you are near a boundary. So, if $i - 1 > 0$ that is if you are not at the left boundary, then you must do something. And that something will be $col \text{ of } ncols \cdot j$ will be equal to j , no problem; $col \text{ of } ncols \cdot i$ will be $i - 1$, and $col \text{ of } ncols \text{ yeah } ncols$ is 1.

So, when we are entering into this particular thing $ncols$ is 1, so we are setting it to. So, we have already set the col_0 . So, we are setting the col_1 . At $col \cdot ncol \cdot i$ will be i minus 1 and value will be 1. Well, $ncols$ will be equal to 1. And in the end, if this is satisfied, then we must increment the $ncol$ by 1 because we will again check whether it is near it is not near some other boundary.

So, if $i + 1$ is $< info \cdot mx - 1$. So, this this is just to check whether you are not near the right boundary. If you are not near the right boundary, then you can set the columns corresponding to that particular boundary as well. So, this will be still j , this will be i plus 1, and this will be 1, and $n \cdot ncol$ will be incremented.

(Refer Slide Time: 46:59)

The image shows a code editor on the left and a documentation page on the right. The code editor displays a C++ snippet for setting values in a matrix. The documentation page is titled "MatSetValuesStencil" and provides a synopsis, synopsis, and input parameters.

```
38 for (i = info.xs; i < (info.xs + info.xm); i++)
39 {
40     row.j = j; row.i = i;
41     col[0].j = j; col[0].i = i;
42
43     ncols = 1;
44
45     if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my - 1)
46     {
47         v[0] = 1.0;
48     }
49     else
50     {
51         v[0] = 4;
52         if (i > 0)
53         {
54             col[ncols].j = j; col[ncols].i = i - 1; val[ncols] =
55             ncol = ncol + 1;
56         }
57         if (i < info.mx - 1)
58         {
59             col[ncols].j = j; col[ncols].i = i + 1; val[ncols] =
60             ncol = ncol + 1;
61         }
62     }
63 }
64
65
```

petsc-3.14.3 2021-01-09
[Report Typos and Errors](#)

MatSetValuesStencil

Inserts or adds a block of values into a matrix. Using structured grid indexing

Synopsis

```
#include "petsc.h"
PetscErrorCode MatSetValuesStencil(Mat mat, PetscInt m, cor
Not Collective
```

Input Parameters

- mat** - the matrix
- m** - number of rows being entered
- idxm** - grid coordinates (and component number when dof > 1) for matrix rows being entered
- n** - number of columns being entered
- idxn** - grid coordinates (and component number when dof > 1) for matrix columns being entered
- v** - a logically two-dimensional array of values

addv - either **ADD_VALUES** or **INSERT_VALUES**. where **ADD_VALUES** adds values to any existing entries, and **INSERT_VALUES** replaces

Then similarly we can copy these two conditionals, and we can paste it.

(Refer Slide Time: 47:03)

The image shows a code editor on the left and a handwritten note on the right. The code editor displays a C++ snippet for setting values in a matrix. The handwritten note explains DMDA (Distributed Memory Data Access) and matrix operations.

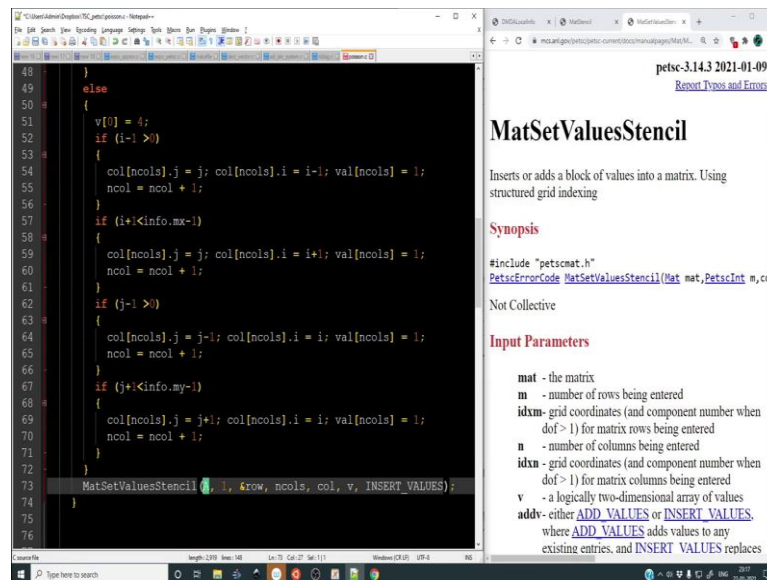
```
48 }
49 else
50 {
51     v[0] = 4;
52     if (i > 0)
53     {
54         col[ncols].j = j; col[ncols].i = i - 1; val[ncols] =
55         ncol = ncol + 1;
56     }
57     if (i < info.mx - 1)
58     {
59         col[ncols].j = j; col[ncols].i = i + 1; val[ncols] =
60         ncol = ncol + 1;
61     }
62     if (j > 0)
63     {
64         col[ncols].j = j; col[ncols].i = i - 1; val[ncols] =
65         ncol = ncol + 1;
66     }
67     if (j < info.my - 1)
68     {
69         col[ncols].j = j; col[ncols].i = i + 1; val[ncols] =
70         ncol = ncol + 1;
71     }
72 }
73
74
75
```

DMDA → structure of u
distributed member distributed array
data management
 $\vec{A} \vec{x} = \vec{b}$ $m \times n$
matrix $(m \times n) \times (n \times 1)$
 $m^1 \times m^2$

$u_d(x, y) = (x^2 - x^4)(y^4 - y^2)$

So, we have to check whether you are not at the lower. Well, not at the lower boundary meaning we are not at. So, if this is the grid just think I should iterate this because the value of u is 0 everywhere. So, if you are at this or rather at this boundary and you need contributions from this neighbour, this neighbour, this neighbour, but not this neighbour because it is 0 alright that is why I am setting it like this ok. So, if you are not at the low boundary, then this will be $j-1$.

(Refer Slide Time: 47:39)



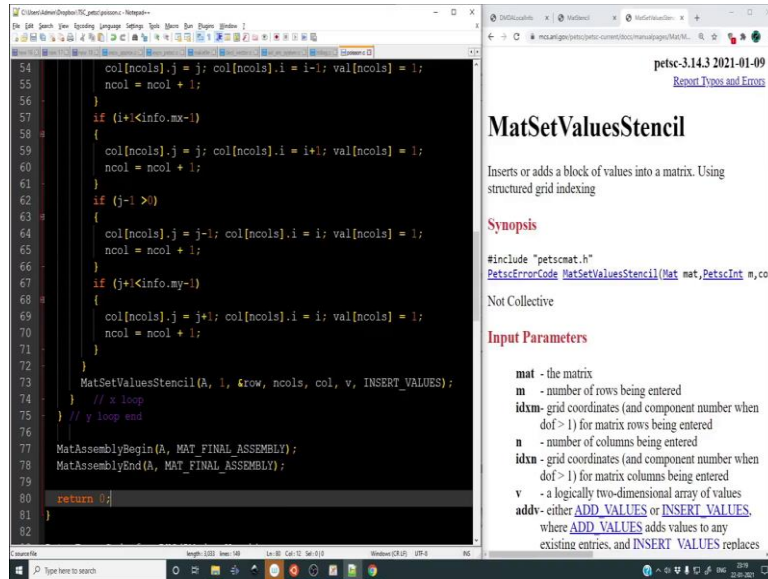
This will be i , and the value will be 1, `ncol` has to be incremented. And lastly, if it is not in the neighbourhood of the top boundary, then we can do this, and we can do this alright. So, so far we have created everything we have accounted for the fact that the value at the boundaries is 0. At the end of this loop, we can now set the matrix.

So, let us take stop of whatever we have done, yeah. We have done this if else at the end of. So, before this ends, you must set. So, before this end, we must set this. So, we will pass it to a, 1, row, and cols, yeah. So, if everything is if all the point in contention, if the i j is inside, then it will assign this definitely, it will assign also this increment `ncol`, it will assign this increment `ncol`, assign this increment `ncol`, assign this increment `ncol`, so you will have 5 such insertions.

If not, if you are at the neighbourhood of the left boundary right, then this will not be executed because you are you have already accounted for the fact that the left boundary condition is equal to 0. If you are in the vicinity of the right boundary, so that the nearest neighbour of the i , j is the right boundary, then you do not you will not go into this statement solely because of this ok.

So, this is how you can account for those end cases alright. So, here we have accounted for the end cases. We have set the values from the stencil alright.

(Refer Slide Time: 49:41)



```
54   col[ncols].j = j; col[ncols].i = i-1; val[ncols] = 1;
55   ncol = ncol + 1;
56   }
57   if (i+1<info.mx-1)
58   {
59   col[ncols].j = j; col[ncols].i = i+1; val[ncols] = 1;
60   ncol = ncol + 1;
61   }
62   if (j-1 >0)
63   {
64   col[ncols].j = j-1; col[ncols].i = i; val[ncols] = 1;
65   ncol = ncol + 1;
66   }
67   if (j+1<info.my-1)
68   {
69   col[ncols].j = j+1; col[ncols].i = i; val[ncols] = 1;
70   ncol = ncol + 1;
71   }
72   }
73   MatSetValuesStencil(A, 1, &row, ncol, col, v, INSERT_VALUES);
74   // x loop
75   } // y loop end
76   }
77   MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
78   MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
79
80   return 0;
81 }
82
```

petsc-3.14.3 2021-01-09

[Report Typos and Errors](#)

MatSetValuesStencil

Inserts or adds a block of values into a matrix. Using structured grid indexing

Synopsis

```
#include "petsc.h"
PetscErrorCode MatSetValuesStencil(Mat mat, PetscInt m, cor
```

Not Collective

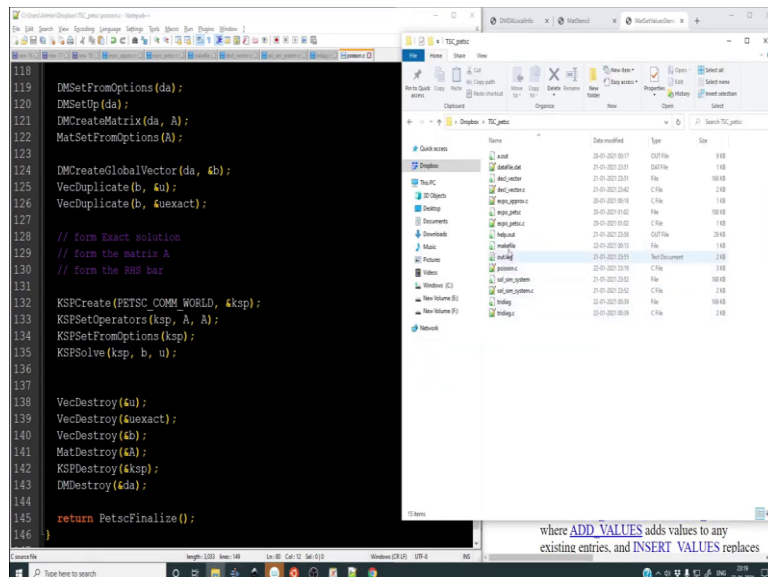
Input Parameters

- mat** - the matrix
- m** - number of rows being entered
- idxm** - grid coordinates (and component number when dof > 1) for matrix rows being entered
- n** - number of columns being entered
- idxn** - grid coordinates (and component number when dof > 1) for matrix columns being entered
- v** - a logically two-dimensional array of values
- adv** - either `ADD_VALUES` or `INSERT_VALUES`, where `ADD_VALUES` adds values to any existing entries, and `INSERT_VALUES` replaces

Then lastly we must do MatAssemblyBegin A comma MAT FINAL ASSEMBLY. And finally, we must end this assembly as we have done before as well End alright. Finally we must return 0, and. So, this assembly has to be done after both the loops are executed. So, this is the end of y loop, this is the end of x loop.

So, you must conclude with this. Let me just indent it. So, it does not really matter. The indentation in C does not really bother us. Finally, we will return 0 if everything is correct ok. So, now we have all of this going on. Let us see if everything compiles.

(Refer Slide Time: 50:45)



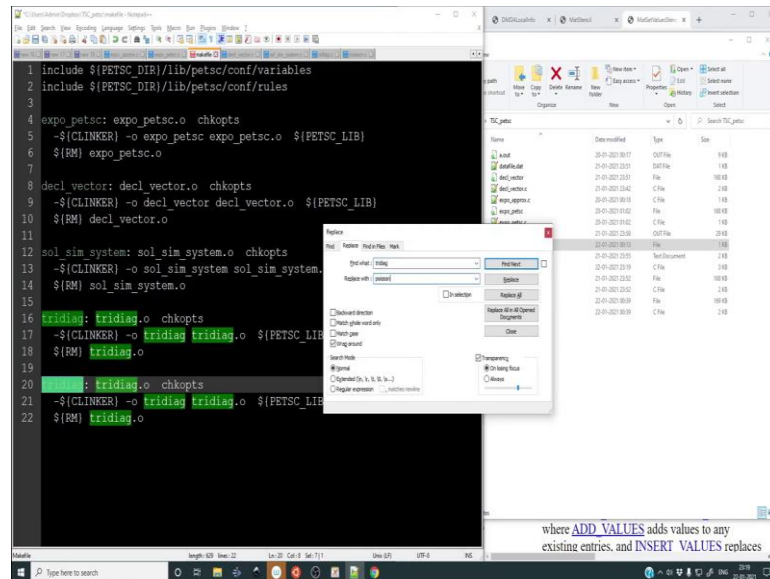
```
118
119 DMSolveFromOptions(da);
120 DMSolve(da);
121 DMCreateMatrix(da, A);
122 MatSetFromOptions(A);
123
124 DMCreateGlobalVector(da, &b);
125 VecDuplicate(b, &u);
126 VecDuplicate(b, &uexact);
127
128 // form exact solution
129 // form the matrix A
130 // form the RHS bar
131
132 KSPCreate(PETSC_COMM_WORLD, &ksp);
133 KSPSetOperators(ksp, A, A);
134 KSPSetFromOptions(ksp);
135 KSPSolve(ksp, b, u);
136
137
138 VecDestroy(&u);
139 VecDestroy(&uexact);
140 VecDestroy(&b);
141 MatDestroy(&A);
142 KSPDestroy(&ksp);
143 DMDestroy(&da);
144
145 return PetscFinalize();
146 }
```

Name	Date modified	Type	Size
add	20-01-2021 10:17	DS7File	1 KB
add_mat	20-01-2021 10:18	DS7File	1 KB
add_vector	20-01-2021 10:42	CFile	2 KB
app_option	20-01-2021 09:16	CFile	1 KB
bd	20-01-2021 01:02	File	368 KB
docs	20-01-2021 01:02	CFile	1 KB
exact	20-01-2021 10:58	DS7File	2 KB
mat	20-01-2021 09:19	File	1 KB
mat_vec	20-01-2021 09:19	CFile	1 KB
mat_vec_systemic	20-01-2021 01:02	CFile	368 KB
mat	20-01-2021 09:19	CFile	1 KB
mat	20-01-2021 09:19	CFile	1 KB

where `ADD_VALUES` adds values to any existing entries, and `INSERT_VALUES` replaces

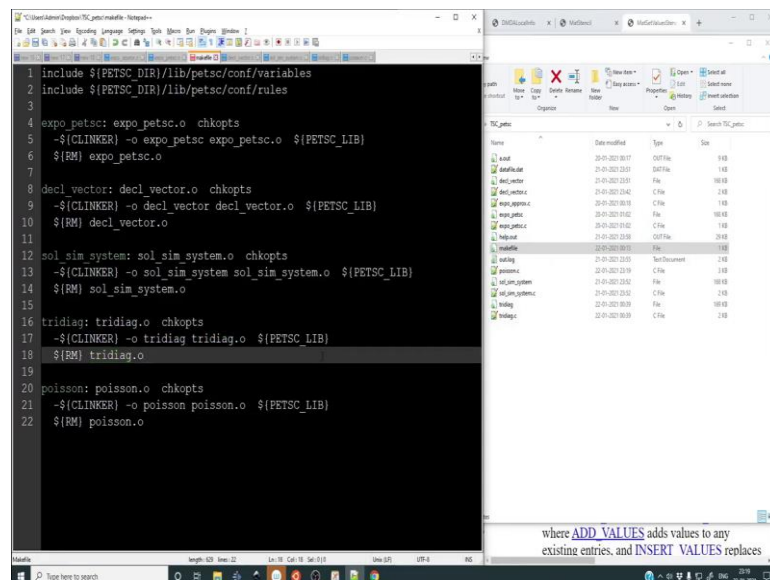
So, for that, let me open up the make file.

(Refer Slide Time: 50:47)



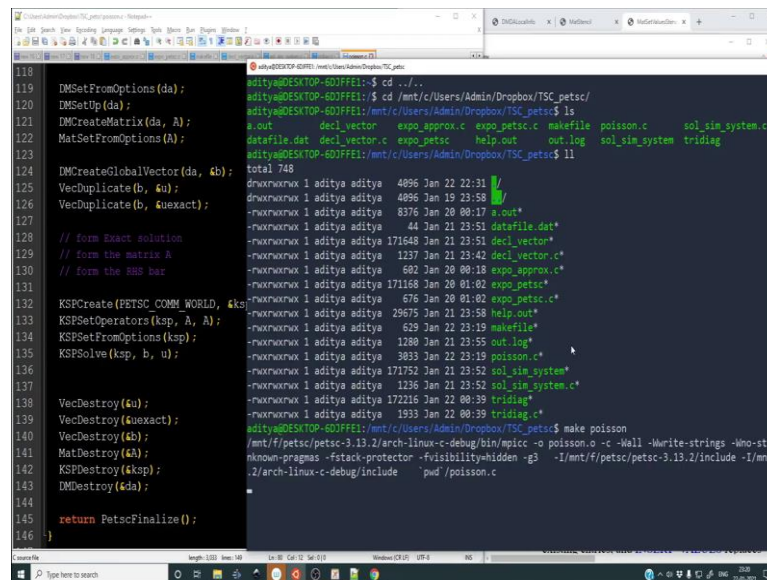
Let me create a new target. So, what is the target? And the name of the file is poisson. Ctrl H, Ctrl H is to replace. So, tridiag has to be replaced by poisson, replace, replace, replace, replace, replace.

(Refer Slide Time: 51:09)



Well, we must make all the shortcuts available to us ok.

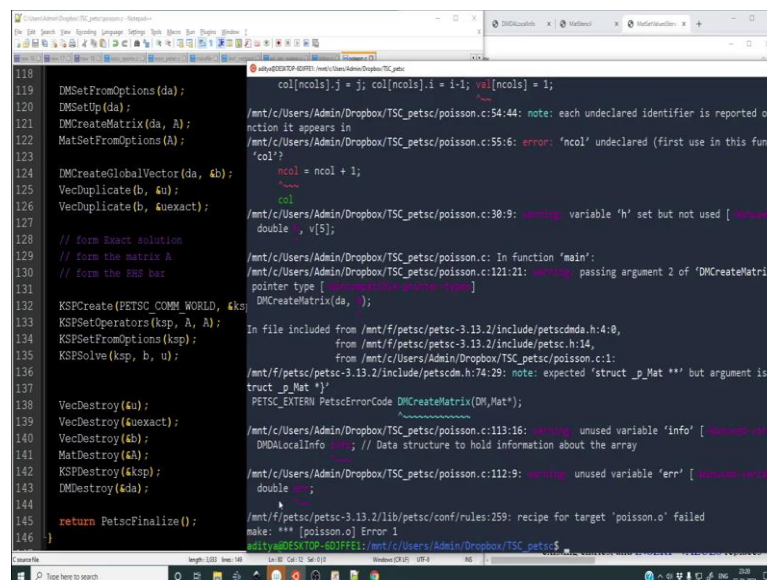
(Refer Slide Time: 51:19)



```
118
119 DMSetFromOptions(da);
120 DMSetUp(da);
121 DMCreateMatrix(da, A);
122 MatSetFromOptions(A);
123
124 DMCreateGlobalVector(da, &b);
125 VecDuplicate(b, &u);
126 VecDuplicate(b, &uexact);
127
128 // form exact solution
129 // form the matrix A
130 // form the RHS bar
131
132 KSPCreate(PETSC_COMM_WORLD, &ksp);
133 KSPSetOperators(ksp, A, A);
134 KSPSetFromOptions(ksp);
135 KSPSolve(ksp, b, u);
136
137
138 VecDestroy(&u);
139 VecDestroy(&uexact);
140 VecDestroy(&b);
141 MatDestroy(&A);
142 KSPDestroy(&ksp);
143 DMDestroy(&da);
144
145 return PetscFinalize();
146 }
```

So, now let us let me just quickly navigate. So, now we have the make file over here. So, make poisson unused variable ok.

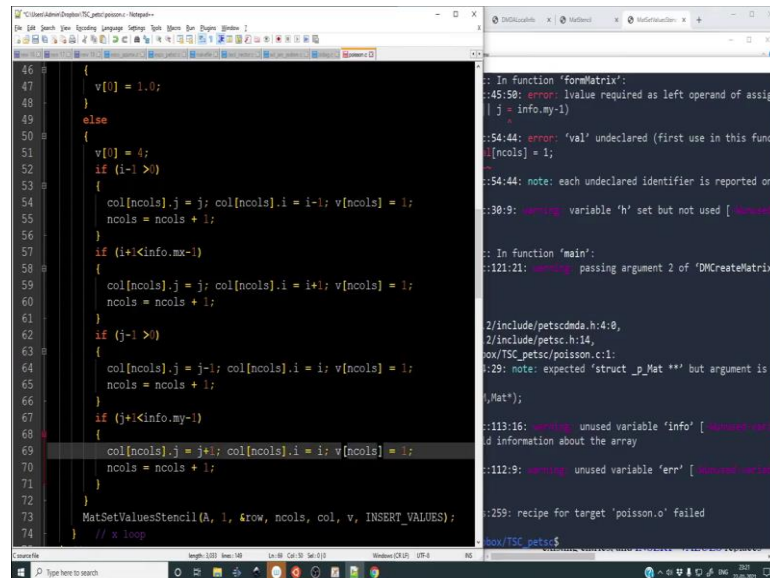
(Refer Slide Time: 51:49)



```
col[ncols].j = j; col[ncols].i = i-1; val[ncols] = 1;
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:54:44: note: each undeclared identifier is reported only once; it appears here
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:55:6: error: 'ncol' undeclared (first use in this function)
      ncol = ncol + 1;
      ^
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:38:9: warning: variable 'h' set but not used [Wunused-but-set-variable]
      double h, v[5];
      ^
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'main':
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:121:21: warning: passing argument 2 of 'DMCreateMatrix' from incompatible pointer type [warning]
DMCreateMatrix(da, );
^
In file included from /mt/f/petsc/petsc-3.13.2/include/petscdmda.h:4:0,
                 from /mt/f/petsc/petsc-3.13.2/include/petsc.h:14,
                 from /mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:1:
/mt/f/petsc/petsc-3.13.2/include/petscdm.h:74:29: note: expected 'struct _p_Mat **' but argument is of type 'p_Mat *'
PETSC_EXTERN PetscErrorCode DMCreateMatrix(DM,Mat*);
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:113:16: warning: unused variable 'info' [Wunused-variable]
DMCMLocalInfo info; // Data structure to hold information about the array
/mt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:112:9: warning: unused variable 'err' [Wunused-variable]
double err;
/mt/f/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
aditya@DESKTOP-6DJFFE1: /mt/c/Users/Admin/Dropbox/TSC_petsc$
```

Unused variable that is fine ok, ncol there seems to be an error in ncol, we have called it ncols. Well, ncols, ncols, ncols, ncols, ncols.

(Refer Slide Time: 52:09)

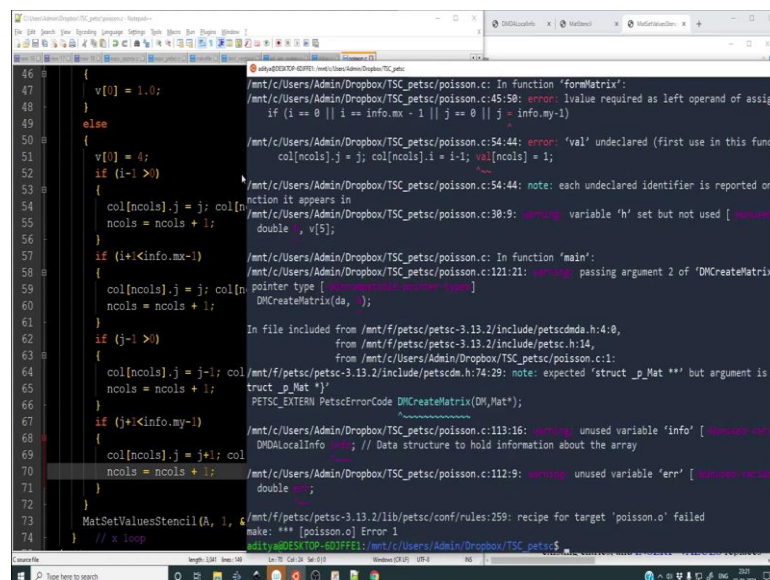


```
46 {
47     v[0] = 1.0;
48 }
49 else
50 {
51     v[0] = 4;
52     if (i-1 > 0)
53     {
54         col[ncols].j = j; col[ncols].i = i-1; v[ncols] = 1;
55         ncols = ncols + 1;
56     }
57     if (i+1 < info.mx-1)
58     {
59         col[ncols].j = j; col[ncols].i = i+1; v[ncols] = 1;
60         ncols = ncols + 1;
61     }
62     if (j-1 > 0)
63     {
64         col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
65         ncols = ncols + 1;
66     }
67     if (j+1 < info.my-1)
68     {
69         col[ncols].j = j+1; col[ncols].i = i; v[ncols] = 1;
70         ncols = ncols + 1;
71     }
72     MatSetValuesStencil(A, 1, &row, ncols, col, v, INSERT_VALUES);
73 } // x loop
```

```
... In function 'formMatrix':
...:45:58: error: lvalue required as left operand of assignment
... | j = info.my-1)
...:54:44: error: 'val' undeclared (first use in this function)
... | col[ncols].j = 1;
...:54:44: note: each undeclared identifier is reported only once
...:30:9: warning: variable 'h' set but not used [Wunused-but-set-variable]
...: In function 'main':
...:121:21: warning: passing argument 2 of 'DMCreateMatrix'
...
.../include/petscdmda.h:4:0,
.../include/petsc.h:14,
.../TSC_petsc/poisson.c:1:
...:29: note: expected 'struct_p_Mat ***' but argument is of type
... (Mat*);
...:113:16: warning: unused variable 'info' [Wunused-variable]
... and information about the array
...:112:9: warning: unused variable 'err' [Wunused-variable]
...:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
```

These are some silly mistakes, which you can clearly figure it out from this. Let me compile again. Let me clear everything and compile, yeah ok.

(Refer Slide Time: 52:27)

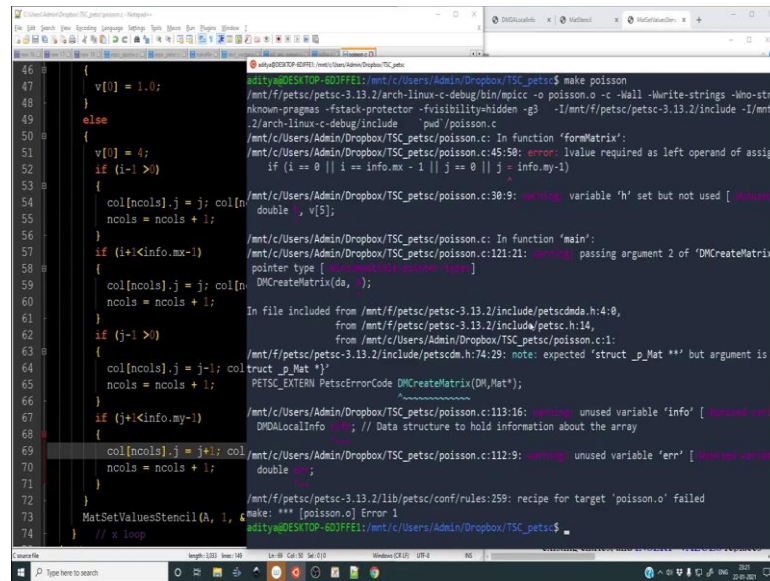


```
46 {
47     v[0] = 1.0;
48 }
49 else
50 {
51     v[0] = 4;
52     if (i-1 > 0)
53     {
54         col[ncols].j = j; col[ncols].i = i-1; val[ncols] = 1;
55         ncols = ncols + 1;
56     }
57     if (i+1 < info.mx-1)
58     {
59         col[ncols].j = j; col[ncols].i = i+1; val[ncols] = 1;
60         ncols = ncols + 1;
61     }
62     if (j-1 > 0)
63     {
64         col[ncols].j = j-1; col[ncols].i = i; val[ncols] = 1;
65         ncols = ncols + 1;
66     }
67     if (j+1 < info.my-1)
68     {
69         col[ncols].j = j+1; col[ncols].i = i; val[ncols] = 1;
70         ncols = ncols + 1;
71     }
72     MatSetValuesStencil(A, 1, &row, ncols, col, v, INSERT_VALUES);
73 } // x loop
```

```
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'formMatrix':
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:45:58: error: lvalue required as left operand of assignment
... if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my-1)
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:54:44: error: 'val' undeclared (first use in this function)
... | col[ncols].j = j; col[ncols].i = i-1; val[ncols] = 1;
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:54:44: note: each undeclared identifier is reported only once
... function it appears in
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:30:9: warning: variable 'h' set but not used [Wunused-but-set-variable]
... double v, v[5];
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'main':
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:121:21: warning: passing argument 2 of 'DMCreateMatrix'
... pointer type [warning: passing argument 2 of 'DMCreateMatrix'
... DMCreateMatrix(da, );
... In file included from /mnt/f/petsc/petsc-3.13.2/include/petscdmda.h:4:0,
... from /mnt/f/petsc/petsc-3.13.2/include/petsc.h:14,
... from /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:1:
... /mnt/f/petsc/petsc-3.13.2/include/petscdm.h:74:29: note: expected 'struct_p_Mat ***' but argument is of type
... struct_p_Mat *)
... PETSC_EXTERN PetscErrorCode DMCreateMatrix(DM,Mat*);
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:113:16: warning: unused variable 'info' [Wunused-variable]
... DMQALocalInfo info; // Data structure to hold information about the array
... /mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:112:9: warning: unused variable 'err' [Wunused-variable]
... double err;
... /mnt/f/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
```

So, val is simply v, I got carried away, no problem, this should be fine. Let me clear everything and compile again.

(Refer Slide Time: 52:41)

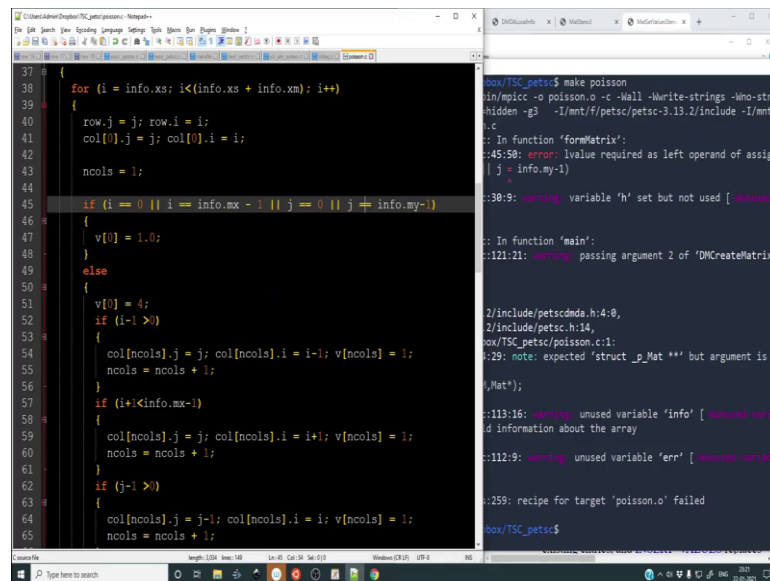


```
46 {
47     v[0] = 1.0;
48 }
49 else
50 {
51     v[0] = 4;
52     if (i-1 > 0)
53     {
54         col[ncols].j = j; col[n
55         ncols = ncols + 1;
56     }
57     if (i+1 < info.mx-1)
58     {
59         col[ncols].j = j; col[n
60         ncols = ncols + 1;
61     }
62     if (j-1 > 0)
63     {
64         col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
65         ncols = ncols + 1;
66     }
67     if (j+1 < info.my-1)
68     {
69         col[ncols].j = j+1; col[ncols].i = i; v[ncols] = 1;
70         ncols = ncols + 1;
71     }
72     MatSetValuesStencil(A, 1,
73     // x loop
74 }
```

```
aditya@DESKTOP-GDJFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make poisson
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strin
known-pragmas -fstack-protector -fvvisibility-hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include -I/mnt/
.2/arch-linux-c-debug/include -pud /poisson.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c: In function 'formMatrix':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:45:58: error: lvalue required as left operand of assign
    if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my-1)
                                                    ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:38:9: warning: variable 'h' set but not used [-Wunused-va
double h, v[5],
         ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:121:21: warning: passing argument 2 of 'DMCreateMatrix'
pointer type [warning: passing argument 2 of 'DMCreateMatrix'
DMCreateMatrix(ds, );
/mnt/ff/petsc/petsc-3.13.2/include/petscdmda.h:4:8,
/mnt/ff/petsc/petsc-3.13.2/include/petsc.h:14,
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:1:
note: expected 'struct_p_Mat ***' but argument is o
PETSC_EXTERN PetscErrorCode DMCreateMatrix(DM,Mat*);
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:113:16: warning: unused variable 'info' [-Wunused-vari
DMDataLocalInfo info; // Data structure to hold information about the array
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:112:9: warning: unused variable 'err' [-Wunused-variab
double err;
/mnt/ff/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
aditya@DESKTOP-GDJFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

This has to be double equal to look.

(Refer Slide Time: 52:47)

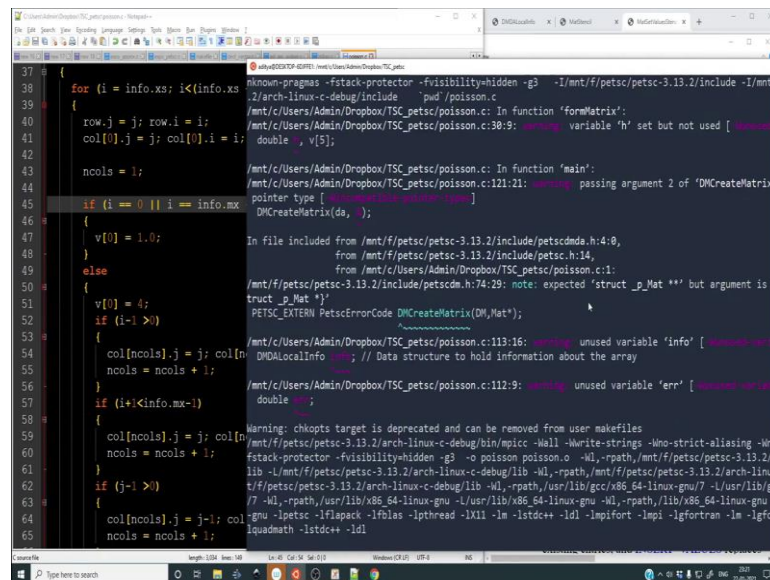


```
37 {
38     for (i = info.xs; i < (info.xs + info.xm); i++)
39     {
40         row.j = j; row.i = i;
41         col[0].j = j; col[0].i = i;
42         ncols = 1;
43     }
44     if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my-1)
45     {
46         v[0] = 1.0;
47     }
48     else
49     {
50         v[0] = 4;
51         if (i-1 > 0)
52         {
53             col[ncols].j = j; col[ncols].i = i-1; v[ncols] = 1;
54             ncols = ncols + 1;
55         }
56         if (i+1 < info.mx-1)
57         {
58             col[ncols].j = j; col[ncols].i = i+1; v[ncols] = 1;
59             ncols = ncols + 1;
60         }
61         if (j-1 > 0)
62         {
63             col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
64             ncols = ncols + 1;
65         }
66     }
67 }
```

```
aditya@DESKTOP-GDJFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make poisson
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strin
known-pragmas -fstack-protector -fvvisibility-hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include -I/mnt/
.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c: In function 'formMatrix':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:45:58: error: lvalue required as left operand of assign
    if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my-1)
                                                    ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:38:9: warning: variable 'h' set but not used [-Wunused-va
double h, v[5],
         ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:121:21: warning: passing argument 2 of 'DMCreateMatrix'
pointer type [warning: passing argument 2 of 'DMCreateMatrix'
DMCreateMatrix(ds, );
/mnt/ff/petsc/petsc-3.13.2/include/petscdmda.h:4:8,
/mnt/ff/petsc/petsc-3.13.2/include/petsc.h:14,
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:1:
note: expected 'struct_p_Mat ***' but argument is o
PETSC_EXTERN PetscErrorCode DMCreateMatrix(DM,Mat*);
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:113:16: warning: unused variable 'info' [-Wunused-vari
DMDataLocalInfo info; // Data structure to hold information about the array
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poison.c:112:9: warning: unused variable 'err' [-Wunused-variab
double err;
/mnt/ff/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
aditya@DESKTOP-GDJFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

These are all errors which you make when you are writing code quickly. Look there is no rush really I mean you can take your time to write a program.

(Refer Slide Time: 52:57)



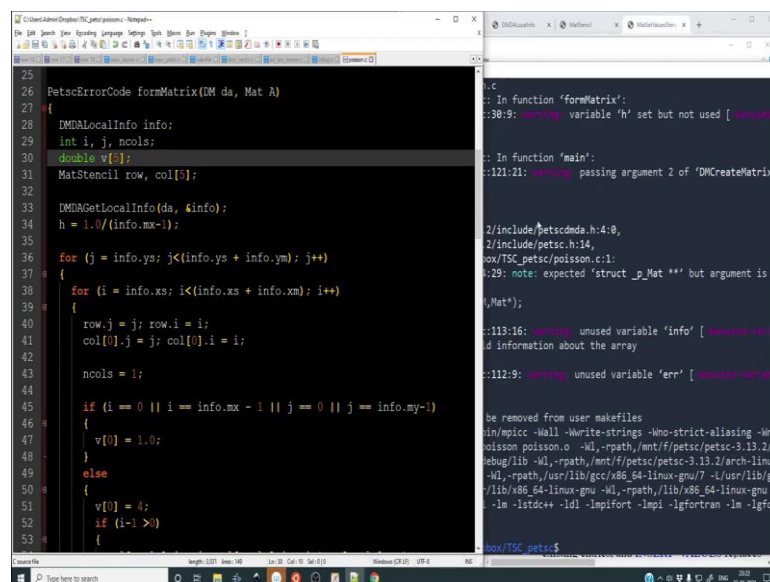
```
37 {
38   for (i = info.xs; i < (info.xs
39   {
40     row.j = j; row.i = i;
41     col[0].j = j; col[0].i = i;
42
43     ncols = 1;
44
45     if (i == 0 || i == info.mx
46     {
47       v[0] = 1.0;
48     }
49     else
50     {
51       v[0] = 4;
52       if (i-1 > 0)
53       {
54         col[ncols].j = j; col[ncols
55         ncols = ncols + 1;
56       }
57       if (i+1 < info.mx-1)
58       {
59         col[ncols].j = j; col[ncols
60         ncols = ncols + 1;
61       }
62       if (j-1 > 0)
63       {
64         col[ncols].j = j-1; col[ncols
65         ncols = ncols + 1;
66     }
67   }
68 }
```

Compiler warnings visible in the background:

- Warning: variable 'h' set but not used
- Warning: passing argument 2 of 'DMCreateMatrix' pointer type
- Warning: unused variable 'info'
- Warning: unused variable 'err'

And now cancel not now ok. So, now, what we have is unused variable, we have not used info in the main. We have not used the error in the main DMCreateMatrix, what is it? Double h not used.

(Refer Slide Time: 53:23)



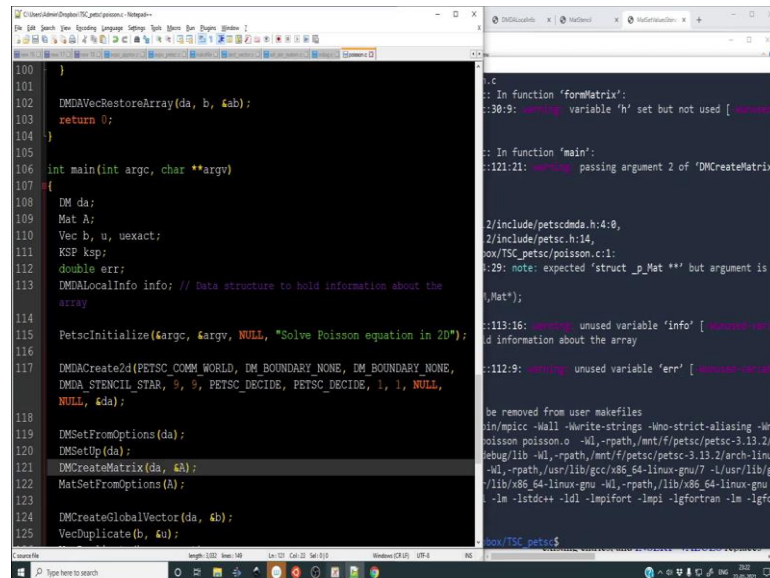
```
26 PetscErrorCode formMatrix(DM da, Mat A)
27 {
28   DMDALocalInfo info;
29   int i, j, ncols;
30   double v[5];
31   MatStencil row, col[5];
32
33   DMDAGetLocalInfo(da, &info);
34   h = 1.0/(info.mx-1);
35
36   for (j = info.ys; j < (info.ys + info.yh); j++)
37   {
38     for (i = info.xs; i < (info.xs + info.xh); i++)
39     {
40       row.j = j; row.i = i;
41       col[0].j = j; col[0].i = i;
42
43       ncols = 1;
44
45       if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my-1)
46       {
47         v[0] = 1.0;
48       }
49       else
50       {
51         v[0] = 4;
52         if (i-1 > 0)
53         {
54           col[ncols].j = j; col[ncols
55           ncols = ncols + 1;
56         }
57         if (i+1 < info.mx-1)
58         {
59           col[ncols].j = j; col[ncols
60           ncols = ncols + 1;
61         }
62         if (j-1 > 0)
63         {
64           col[ncols].j = j-1; col[ncols
65           ncols = ncols + 1;
66         }
67       }
68     }
69   }
70 }
```

Compiler warnings visible in the background:

- Warning: variable 'h' set but not used
- Warning: passing argument 2 of 'DMCreateMatrix' pointer type
- Warning: unused variable 'info'
- Warning: unused variable 'err'

Well, double, yeah this is not used. So, you can remove that. DMCreateMatrix. Where is it? DMCreateMatrix, line 121.

(Refer Slide Time: 53:53)

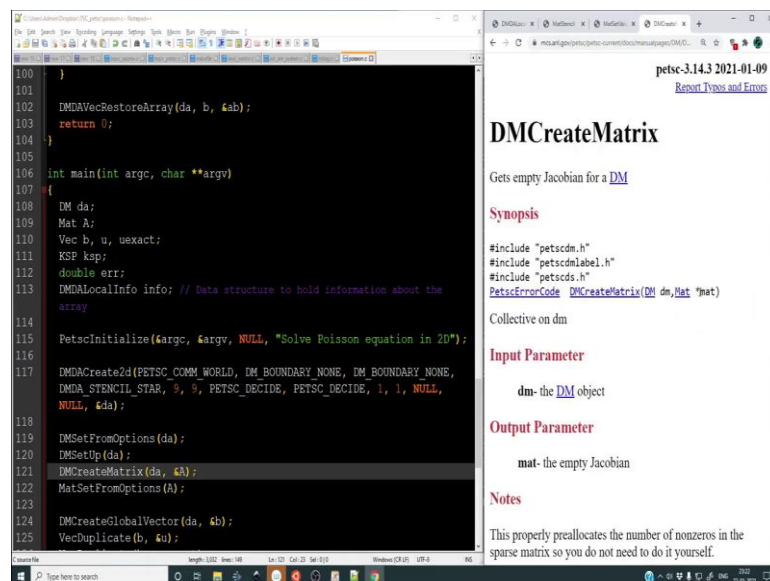


```
100 }
101
102 DMDVecRestoreArray(da, b, &ab);
103 return 0;
104 }
105
106 int main(int argc, char **argv)
107 {
108     DM da;
109     Mat A;
110     Vec b, u, uexact;
111     KSP ksp;
112     double err;
113     DMDLocalInfo info; // Data structure to hold information about the
114     // array
115     PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
116
117     DMDCreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE,
118     DMDA_STENCIL_STAR, 9, 9, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL,
119     NULL, &da);
120
121     DMSetFromOptions(da);
122     DMSetUp(da);
123     DMCCreateMatrix(da, &A);
124     MatSetFromOptions(A);
125
126     DMCCreateGlobalVector(da, &b);
127     VecDuplicate(b, &u);
128 }
```

```
...
: In function 'formMatrix':
:30:9: warning: variable 'h' set but not used [Wunused-but-set-variable]
...
: In function 'main':
:121:21: warning: passing argument 2 of 'DMCreateMatrix'
...
2/include/petscdmda.h:4:8:
2/include/petsc.h:14:
nox/TSC/petsc/poisson.c:1:
:29: note: expected 'struct _p_Mat ***' but argument is of
type 'Mat*';
...
:113:16: warning: unused variable 'info' [Wunused-variable]
...
:112:9: warning: unused variable 'err' [Wunused-variable]
...
be removed from user makefiles
in/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno
poisson poisson.o -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/a
r/lib/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -ldtcd++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
nox/TSC/petsc$
```

So, what is the issue? We must pass the address of A rather than this.

(Refer Slide Time: 54:09)



```
100 }
101
102 DMDVecRestoreArray(da, b, &ab);
103 return 0;
104 }
105
106 int main(int argc, char **argv)
107 {
108     DM da;
109     Mat A;
110     Vec b, u, uexact;
111     KSP ksp;
112     double err;
113     DMDLocalInfo info; // Data structure to hold information about the
114     // array
115     PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
116
117     DMDCreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE,
118     DMDA_STENCIL_STAR, 9, 9, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL,
119     NULL, &da);
120
121     DMSetFromOptions(da);
122     DMSetUp(da);
123     DMCCreateMatrix(da, &A);
124     MatSetFromOptions(A);
125
126     DMCCreateGlobalVector(da, &b);
127     VecDuplicate(b, &u);
128 }
```

petsc-3.14.3 2021-01-09
[Report Typos and Errors](#)

DMCreateMatrix

Gets empty Jacobian for a [DM](#)

Synopsis

```
#include "petscdm.h"
#include "petscdmabel.h"
#include "petscdm.h"
PetscErrorCode DMCCreateMatrix(DM dm,Mat *mat)
```

Collective on dm

Input Parameter

dm- the [DM](#) object

Output Parameter

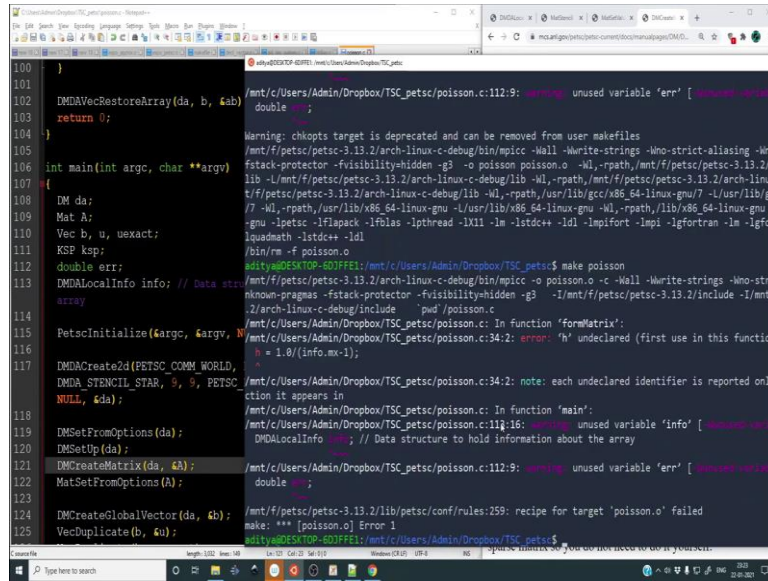
mat- the empty Jacobian

Notes

This properly preallocates the number of nonzeros in the sparse matrix so you do not need to do it yourself.

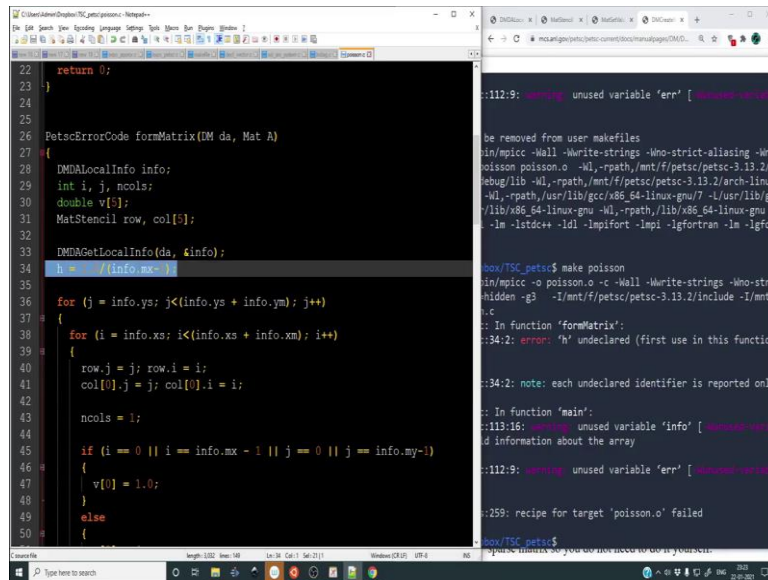
I will show you the functional reference, DMCCreateMatrix, it will be the address of A, not A ok. It is star mat meaning we have to pass the address of the matrix ok. Some silly mistakes removed. It just goes to show that. Which is unused?

(Refer Slide Time: 54:27)



What is it? Info matrix you have made a lot of errors, but that is to err is human ok. Info matrix, we do not need this anymore obviously.

(Refer Slide Time: 54:41)



(Refer Slide Time: 54:45)

```

22     return 0;
23 }
24
25 PetscErrorCode formMatrix(DM da,
26                          DMDALocalInfo info,
27                          MatStencil row, col[]) {
28     DMDALocalInfo info;
29     int i, j, ncols;
30     double v[5];
31     MatStencil row, col[5];
32
33     DMDAGetLocalInfo(da, &info);
34     for (j = info.ys; j < (info.ys +
35                          info.nrows); j++) {
36         for (i = info.xs; i < (info.xs +
37                              info.ncols); i++) {
38             row.j = j; row.i = i;
39             col[0].j = j; col[0].i = i;
40             ncols = 1;
41
42             if (i == 0 || i == info.mx)
43                 v[0] = 1.0;
44             else
45                 v[0] = 4;
46         }
47     }
48     MatSetStencil(row, col, ncols);
49 }
50
51 PetscErrorCode solvePoisson(DM da,
52                             DMDALocalInfo info,
53                             MatStencil row, col[]) {
54     Mat A;
55     Vec u, uexact;
56     VecDuplicate(&uexact);
57     VecDuplicate(uexact, &u);
58     // form Exact solution
59     // form the matrix A
60     // form the RHS bar
61
62     KSPCreate(PETSC_COMM_WORLD, &ksp);
63     KSPSetOperators(ksp, A, A);
64     KSPSetFromOptions(ksp);
65     KSPSolve(ksp, b, u);
66
67     VecDestroy(&u);
68     VecDestroy(&uexact);
69     VecDestroy(&b);
70     MatDestroy(&A);
71     KSPDestroy(&ksp);
72     DMDestroy(&da);
73
74     return PetscFinalize();
75 }

```

Now, we should have only unused warnings great. So, we have successfully debugged our little code and we have removed whatever errors we have encountered.

(Refer Slide Time: 54:57)

```

118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126
127 // form Exact solution
128 // form the matrix A
129 // form the RHS bar
130
131 KSPCreate(PETSC_COMM_WORLD, &ksp);
132 KSPSetOperators(ksp, A, A);
133 KSPSetFromOptions(ksp);
134 KSPSolve(ksp, b, u);
135
136
137 VecDestroy(&u);
138 VecDestroy(&uexact);
139 VecDestroy(&b);
140 MatDestroy(&A);
141 KSPDestroy(&ksp);
142 DMDestroy(&da);
143
144 return PetscFinalize();
145 }
146

```

So, in the end, it is solving. And now once it is solved, how do we know it is correct? We must find out the error. So, again let us do that. So, first things first we must do u minus u_{exact} like in the last exam last lecture $A X + Y$ we must pass u minus 1 and u_{exact} . We cannot like python, we cannot do simply u minus u_{exact} that is not how we can operate

(Refer Slide Time: 57:03)1

```
118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122 double *b;
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126 // form Exact solution
127 // form the matrix A
128 // form the RHS b
129
130
131 KSPCreate(PETSC_COMM_WORLD, &ksp);
132 KSPSetOperators(ksp, A, A);
133 KSPSetFromOptions(ksp);
134 KSPSolve(ksp, b, u);
135
136 VecXPY(u, -1, uexact); // asy
137 VecNorm(u, NORM_INFINITY, &err);
138
139 PetscPrintf(PETSC_COMM_WORLD,
140             "
141             VecDestroy(&u);
142             VecDestroy(&uexact);
143             VecDestroy(&b);
144             MatDestroy(&A);
145             KSPDestroy(&ksp);
146             DMDestroy(&da);
147
148             return PetscFinalize();
149         }
150     }
151     error: inf
152
```

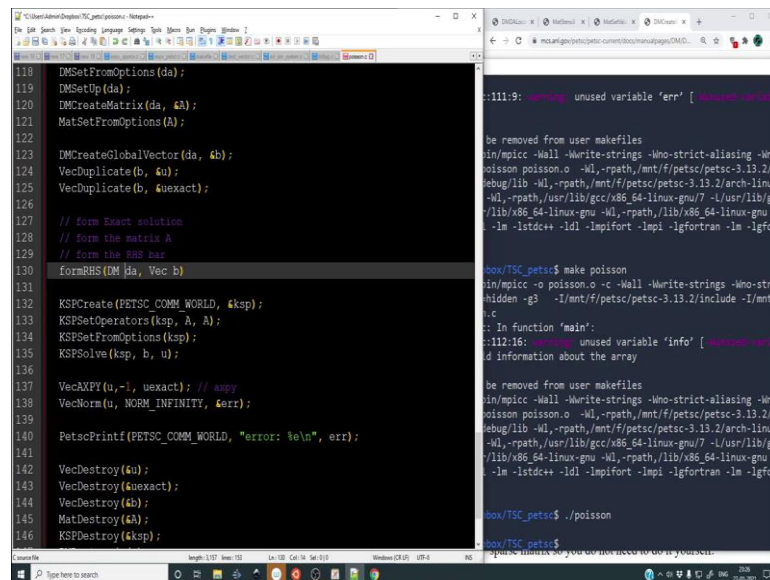
This DMDA info is ok. So, we do not care in the main we have declared this, but we have not used it anyway it does not matter.

(Refer Slide Time: 57:15)

```
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126 double *b;
127 // form Exact solution
128 // form the matrix A
129 // form the RHS b
130
131 KSPCreate(PETSC_COMM_WORLD, &ksp);
132 KSPSetOperators(ksp, A, A);
133 KSPSetFromOptions(ksp);
134 KSPSolve(ksp, b, u);
135
136 VecXPY(u, -1, uexact); // asy
137 VecNorm(u, NORM_INFINITY, &err);
138
139 PetscPrintf(PETSC_COMM_WORLD,
140             "
141             VecDestroy(&u);
142             VecDestroy(&uexact);
143             VecDestroy(&b);
144             MatDestroy(&A);
145             DMDestroy(&da);
146
147             return PetscFinalize();
148         }
149     }
150     error: inf
151     error: inf
152
```

So, after running this dot slash poisson, it will run it with a default 9 by 9 grid, allow access. And it shows infinity. Well, clearly there is something wrong.

(Refer Slide Time: 57:39)

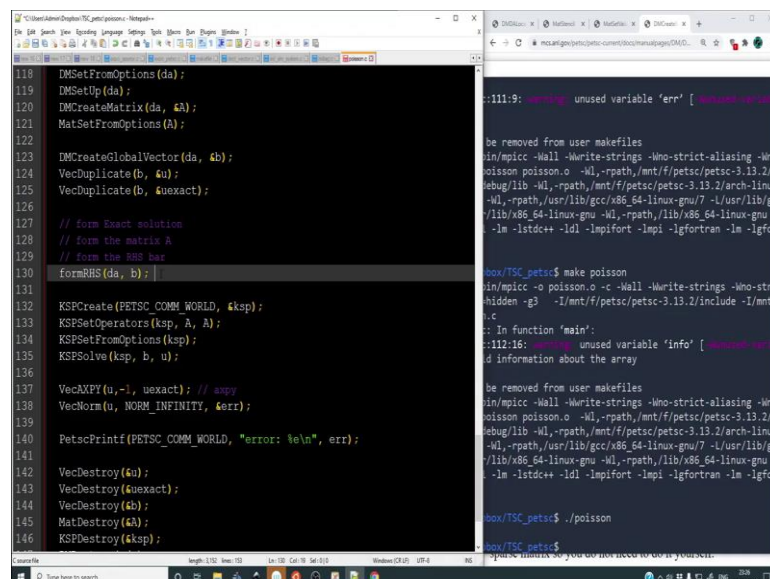


```
118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126
127 // form exact solution
128 // form the matrix A
129 // form the RHS b
130 formRHS(DM da, Vec b)
131
132 KSPCreate(PETSC_COMM_WORLD, &ksp);
133 KSPSetOperators(ksp, A, A);
134 KSPSetFromOptions(ksp);
135 KSPSolve(ksp, b, u);
136
137 VecAXPY(u,-1, uexact); // axpy
138 VecNorm(u, NORM_INFINITY, &err);
139
140 PetscPrintf(PETSC_COMM_WORLD, "error: %e\n", err);
141
142 VecDestroy(&u);
143 VecDestroy(&uexact);
144 VecDestroy(&b);
145 MatDestroy(&A);
146 KSPDestroy(&ksp);
```

```
..111:9: warning: unused variable 'err' [ -Werror=unused-variable ]
be removed from user makefiles
in/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-poission poisson.o -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ make poission
in/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strict-
hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include -I/mnt/
h.c
t: In function 'main':
..112:16: warning: unused variable 'info' [ -Werror=unused-
information-about-the-array ]
be removed from user makefiles
in/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-poission poisson.o -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ ./poisson
box/TSC_petsc$
```

So, let us we have not even called the functions and we have run the program. So, we need to call the functions ok. So, form RHS this, so we need to pass da and the vector b. Then what did we have the matrix form Matrix da , A. Create exact. What is it? What was the function name? uexact 1, u sorry u exact da , uexact alright.

(Refer Slide Time: 58:31)

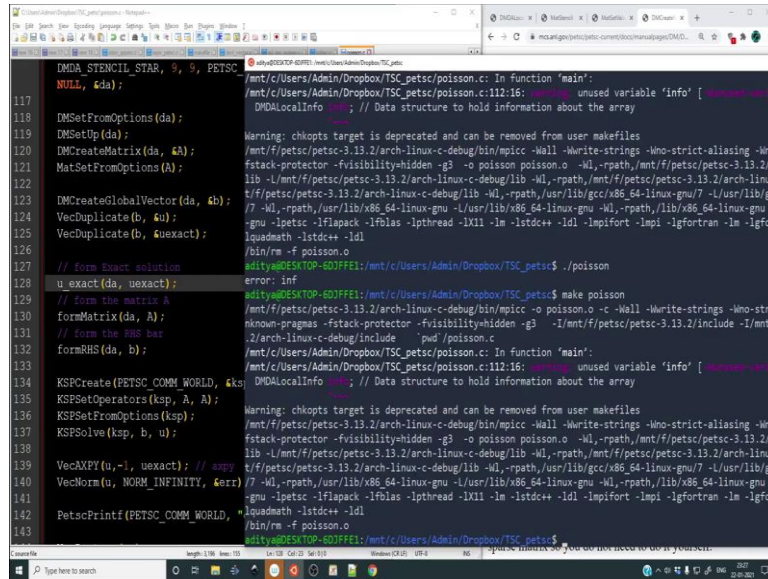


```
118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126
127 // form exact solution
128 // form the matrix A
129 // form the RHS b
130 formRHS(da, b);
131
132 KSPCreate(PETSC_COMM_WORLD, &ksp);
133 KSPSetOperators(ksp, A, A);
134 KSPSetFromOptions(ksp);
135 KSPSolve(ksp, b, u);
136
137 VecAXPY(u,-1, uexact); // axpy
138 VecNorm(u, NORM_INFINITY, &err);
139
140 PetscPrintf(PETSC_COMM_WORLD, "error: %e\n", err);
141
142 VecDestroy(&u);
143 VecDestroy(&uexact);
144 VecDestroy(&b);
145 MatDestroy(&A);
146 KSPDestroy(&ksp);
```

```
..111:9: warning: unused variable 'err' [ -Werror=unused-variable ]
be removed from user makefiles
in/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-poission poisson.o -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ make poission
in/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strict-
hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include -I/mnt/
h.c
t: In function 'main':
..112:16: warning: unused variable 'info' [ -Werror=unused-
information-about-the-array ]
be removed from user makefiles
in/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-poission poisson.o -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ ./poisson
box/TSC_petsc$
```

So, now let me recompile and let me run this ok.

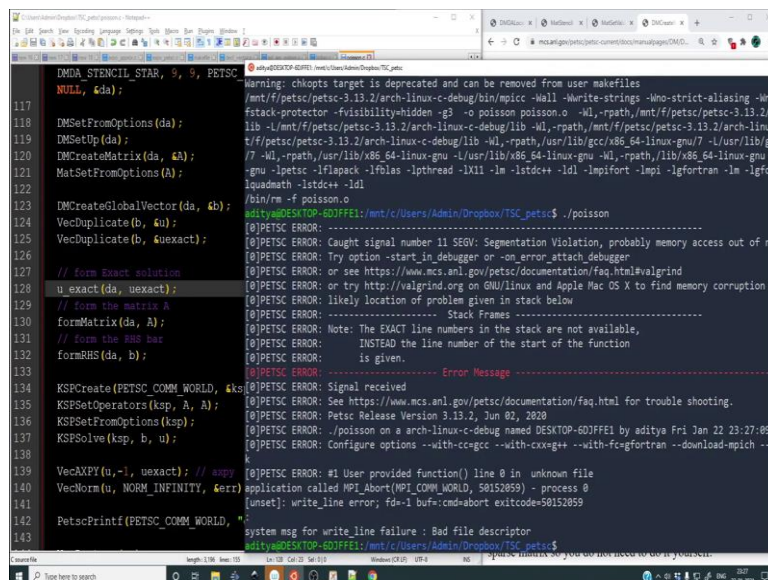
(Refer Slide Time: 58:35)



```
DMDA_STENCIL_STAR, 9, 9, PETSC
NULL, &da);
117
118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126
127 // form exact solution
128 u_exact(da, uexact);
129 // form the matrix A
130 formMatrix(da, A);
131 // form the RHS b
132 formRHS(da, b);
133
134 KSPCreate(PETSC_COMM_WORLD, &ksp);
135 KSPSetOperators(ksp, A, A);
136 KSPSetFromOptions(ksp);
137 KSPSolve(ksp, b, u);
138
139 VecAXPY(u, -1, uexact); // copy
140 VecNorm(u, NORM_INFINITY, &err);
141 PetscPrintf(PETSC_COMM_WORLD,
142
143
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:112:16: warning: unused variable 'info' [ -Werror=unused-variable]
DMDALocalInfo info; // Data structure to hold information about the array
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-fstack-protector -fvisibility=hidden -g3 -o poisson poisson.o -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-t/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -gnu -lpetsc -lflapack -lfdlas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfortquadmath -lstdc++ -ldl
/bin/rm -f poisson.o
aditya@DESKTOP-6D7FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson
error: inf
aditya@DESKTOP-6D7FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make poisson
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-fstack-protector -fvisibility=hidden -g3 -o poisson poisson.o -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-t/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -gnu -lpetsc -lflapack -lfdlas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfortquadmath -lstdc++ -ldl
/bin/rm -f poisson.o
aditya@DESKTOP-6D7FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, there appears to be some error, bad file descriptor. So, there is an out of range issue. Let us see where we can we can whether we can catch it. So, the segmentation fault is appearing somewhere.

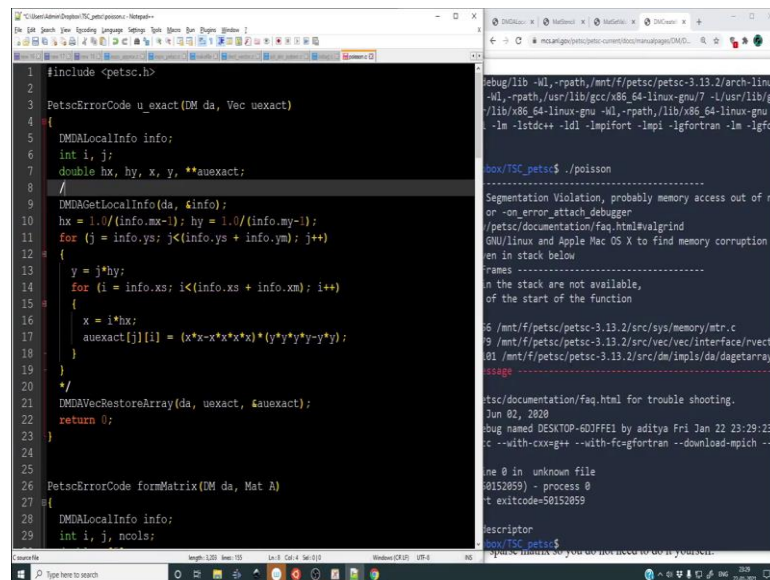
(Refer Slide Time: 59:13)



```
DMDA_STENCIL_STAR, 9, 9, PETSC
NULL, &da);
117
118 DMSetFromOptions(da);
119 DMSetUp(da);
120 DMCreateMatrix(da, &A);
121 MatSetFromOptions(A);
122
123 DMCreateGlobalVector(da, &b);
124 VecDuplicate(b, &u);
125 VecDuplicate(b, &uexact);
126
127 // form exact solution
128 u_exact(da, uexact);
129 // form the matrix A
130 formMatrix(da, A);
131 // form the RHS b
132 formRHS(da, b);
133
134 KSPCreate(PETSC_COMM_WORLD, &ksp);
135 KSPSetOperators(ksp, A, A);
136 KSPSetFromOptions(ksp);
137 KSPSolve(ksp, b, u);
138
139 VecAXPY(u, -1, uexact); // copy
140 VecNorm(u, NORM_INFINITY, &err);
141 PetscPrintf(PETSC_COMM_WORLD,
142
143
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-fstack-protector -fvisibility=hidden -g3 -o poisson poisson.o -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-t/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -gnu -lpetsc -lflapack -lfdlas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfortquadmath -lstdc++ -ldl
/bin/rm -f poisson.o
aditya@DESKTOP-6D7FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson
[0]PETSC ERROR: Caught signal number 11 SEGV: Segmentation Violation, probably memory access out of range
[0]PETSC ERROR: Try option -start_in_debugger or -on_error_attach_debugger
[0]PETSC ERROR: or see https://www.mcs.anl.gov/petsc/documentation/faq.html#valgrind
[0]PETSC ERROR: or try http://valgrind.org on GNU/Linux and Apple Mac OS X to find memory corruption errors
[0]PETSC ERROR: likely location of problem given in stack below
[0]PETSC ERROR: ----- Stack Frames -----
[0]PETSC ERROR: Note: The EXACT line numbers in the stack are not available,
[0]PETSC ERROR: INSTEAD the line number of the start of the function
[0]PETSC ERROR: is given.
[0]PETSC ERROR: ----- Error Message -----
[0]PETSC ERROR: Signal received
[0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
[0]PETSC ERROR: ./poisson on a arch-linux-c-debug named DESKTOP-6D7FFE1 by aditya Fri Jan 22 23:27:09
[0]PETSC ERROR: Configure options --with-ccgcc --with-cxxg++ --with-fcfortran --download-mpich --download-mpi
[0]PETSC ERROR: #1 User provided function() line 0 in unknown file
application called MPI_Abort(MPI_COMM_WORLD, 50152859) - process 0
[0]set[0]: write_line_error; fd=1 buf=cmd:abort exitcode=50152859
system msg for write_line failure : Bad file descriptor
aditya@DESKTOP-6D7FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

Let me comment on this to see whether we can isolate it, whether it is happening over here or not ok.

(Refer Slide Time: 59:53)

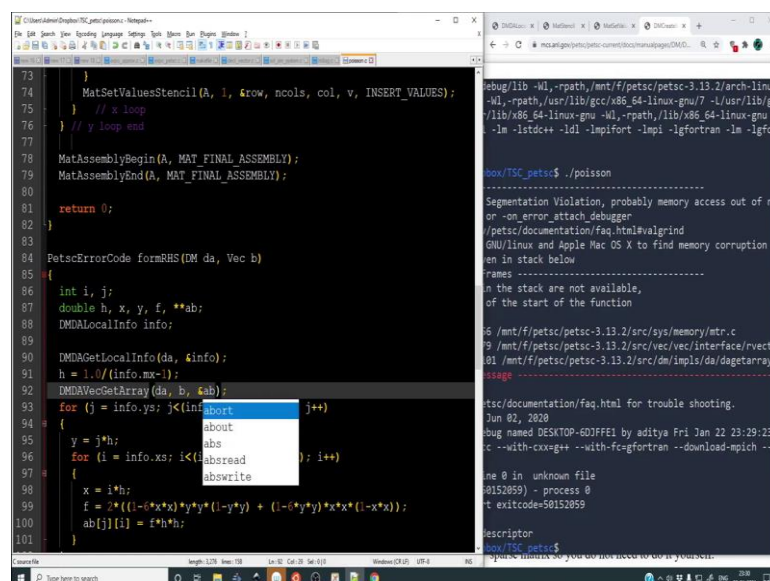


```
1 #include <petsc.h>
2
3 PetscErrorCode u_exact(DM da, Vec uexact)
4 {
5     DMDALocalInfo info;
6     int i, j;
7     double hx, hy, x, y, **auxexact;
8
9     DMDAGetLocalInfo(da, &info);
10    hx = 1.0/(info.mx-1); hy = 1.0/(info.my-1);
11    for (j = info.ys; j<(info.ys + info.ym); j++)
12    {
13        y = j*hy;
14        for (i = info.xs; i<(info.xs + info.xm); i++)
15        {
16            x = i*hx;
17            auxexact[j][i] = (x*x-x*x*x*x)*(y*y*y-y-y-y);
18        }
19    }
20 }
21
22 PetscErrorCode formMatrix(DM da, Vec uexact, &auxexact)
23 {
24     return 0;
25 }
26
27 PetscErrorCode formMatrix(DM da, Mat A)
28 {
29     DMDALocalInfo info;
30     int i, j, ncol;
31 }
```

```
debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ ./poisson
Segmentation Violation, probably memory access out of ra
on -on_error_attach_debugger
/petsc/documentation/faq.html#valgrind
GNU/Linux and Apple Mac OS X to find memory corruption e
an in stack below
frames -----
in the stack are not available,
of the start of the function
6 /mnt/f/petsc/petsc-3.13.2/src/sys/memory/mtr.c
9 /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/rvector
01 /mnt/f/petsc/petsc-3.13.2/src/dm/impls/da/dagetarray-
stage
tsic/documentation/faq.html for trouble shooting.
Jun 02, 2020
bug named DESKTOP-60JFFE1 by aditya Fri Jan 22 23:29:23
c --with-cxxg++ --with-fcgfortran --download-mpich --d
line 0 in unknown file
0152859) - process 0
t exitcode=50152859
descriptor
box/TSC_petsc$
Space: think you go to the next slide to do it yourself.
```

This is a very silly mistake that we did. Before going into this loop, we should have told that we want to link uexact with the auxiliary vector that we want to define on the grid. So, we must definitely do this particular line, this is quite important. So, DMDAVecGetArray, and that is why it was throwing a segmentation fault because it does not know what to do with the pointer, it is simply lost. So, da uexact and the address of auxexact alright. Similarly, we must do this for this thing as well.

(Refer Slide Time: 60:49)



```
73 }
74 MatSetValuesStencil(A, 1, &row, ncol, col, v, INSERT_VALUES);
75 } // x loop
76 } // y loop end
77
78 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
79 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
80
81 return 0;
82 }
83
84 PetscErrorCode formBHS(DM da, Vec b)
85 {
86     int i, j;
87     double h, x, y, f, **ab;
88     DMDALocalInfo info;
89
90     DMDAGetLocalInfo(da, &info);
91     h = 1.0/(info.mx-1);
92     DMDAVecGetArray(da, b, &ab);
93     for (j = info.ys; j<(info.ys + info.ym); j++)
94     {
95         y = j*h;
96         for (i = info.xs; i<(info.xs + info.xm); i++)
97         {
98             absread
99             abswrite
100             x = i*h;
101             f = 2*((1-(x*x))*y*y*(1-y*y) + (1-(y*y))*x*x*(1-x*x));
102             ab[j][i] = f*h*h;
103         }
104     }
105 }
```

```
debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux
-Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc
/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -
-lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgfor
box/TSC_petsc$ ./poisson
Segmentation Violation, probably memory access out of ra
on -on_error_attach_debugger
/petsc/documentation/faq.html#valgrind
GNU/Linux and Apple Mac OS X to find memory corruption e
an in stack below
frames -----
in the stack are not available,
of the start of the function
6 /mnt/f/petsc/petsc-3.13.2/src/sys/memory/mtr.c
9 /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/rvector
01 /mnt/f/petsc/petsc-3.13.2/src/dm/impls/da/dagetarray-
stage
tsic/documentation/faq.html for trouble shooting.
Jun 02, 2020
bug named DESKTOP-60JFFE1 by aditya Fri Jan 22 23:29:23
c --with-cxxg++ --with-fcgfortran --download-mpich --d
line 0 in unknown file
0152859) - process 0
t exitcode=50152859
descriptor
box/TSC_petsc$
Space: think you go to the next slide to do it yourself.
```

So, over here it will be b and ampersand ab, because otherwise it does not know how to because once we are restoring we have to get that as well. And we have seen this in the one of the previous lectures as well ok. Have we done it for the matrix ok? Let us see whether it compiles.

(Refer Slide Time: 61:15)

```

25 }
26
27 PetscErrorCode formMatrix(DM da,
28 int i, j, ncol:
29 double v[5]:
30 MatStencil row, col[1];
31
32 DMDAGetLocalInfo(da, &info);
33
34 for (j = info.ys; j < (info.ys +
35 for (i = info.xs; i < (info.xs
36 {
37     row.j = j; row.i = i;
38     col[0].j = j; col[0].i = i;
39     ncol: = 1;
40     if (i == 0 || i == info.mx
41     {
42         v[0] = 1.0;
43     }
44     else
45     {
46         v[0] = 4;
47         if (i-1 > 0)
48     }
49 }
50 }
51 }
52 }
53 }

```

[0]PETSC ERROR: Note: The EXACT line numbers in the stack are not available,
[0]PETSC ERROR: INSTEAD the line number of the start of the function
[0]PETSC ERROR: is given.
[0]PETSC ERROR: [0] PetscTrFreeDefault line 256 /mnt/f/petsc/petsc-3.13.2/src/sys/memory/mtr.c
[0]PETSC ERROR: [0] VecRestoreArray2d line 2779 /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/r/vecro
[0]PETSC ERROR: [0] DMDAVecRestoreArray line 101 /mnt/f/petsc/petsc-3.13.2/src/dm/impls/da/da/getarray.
----- Error Message -----
[0]PETSC ERROR: Signal received
[0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
[0]PETSC ERROR: ./poisson on a arch-linux-c-debug named DESKTOP-60JFFE1 by aditya Fri Jan 22 23:29:23
[0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc-gfortran --download-mpich --d
[0]PETSC ERROR: #1 User provided function() line 0 in unknown file
application called MPI_Abort(MPI_COMM_WORLD, 50152859) - process 0
[unset]: write_line error; fde-1 buf:cnd=abort exitcode=50152859
system msg for write_line failure : Bad file descriptor
aditya@DESKTOP-60JFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc\$ make poisson
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-stri
known-pragmas -fstack-protector -fvvisibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/
2/arch-linux-c-debug/include -pud/poisson.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'u_exact':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:22:3: error: expected expression before '/' token
*/
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/poisson.c:115:16: warning: unused variable 'info' [-Wunused-vari
DMDALocalInfo info; // Data structure to hold information about the array
/mnt/f/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'poisson.o' failed
make: *** [poisson.o] Error 1
aditya@DESKTOP-60JFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc\$

There is some yeah, ok, forgot about this.

(Refer Slide Time: 61:23)

```

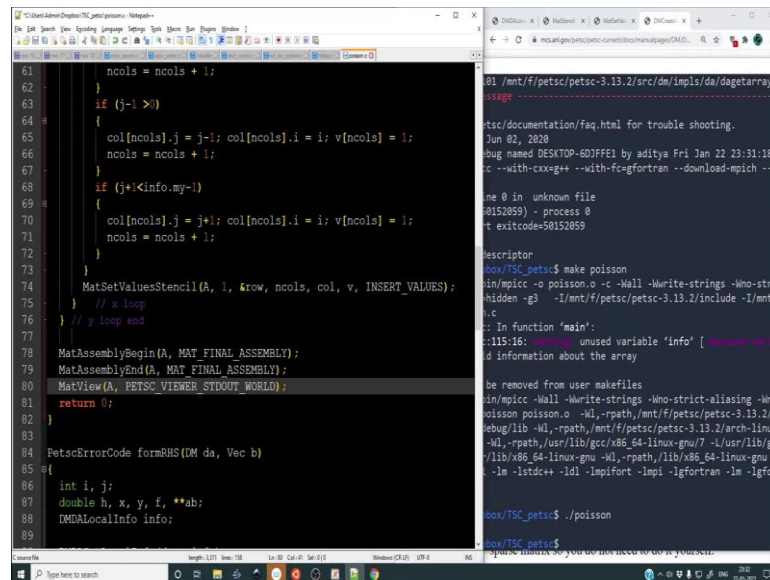
4 {
5     DMDALocalInfo info;
6     int i, j;
7     double hx, hy, x, y, **auexact;
8
9     DMDAGetLocalInfo(da, &info);
10    DMDAVecGetArray(da, uexact, &auexact);
11
12    hx = 1.0/(info.mx-1); hy = 1.0/(info.my-1);
13    for (j = info.ys; j < (info.ys + info.ym); j++)
14    {
15        y = j*hy;
16        for (i = info.xs; i < (info.xs + info.xm); i++)
17        {
18            x = i*hx;
19            auexact[j][i] = (x*x-x*x*x*x)*(y*y*y*y-y*y);
20        }
21    }
22
23    DMDAVecRestoreArray(da, uexact, &auexact);
24    return 0;
25 }
26
27 PetscErrorCode formMatrix(DM da, Mat A)
28 {
29     DMDALocalInfo info;
30     int i, j, ncol:;
31     double v[5];

```

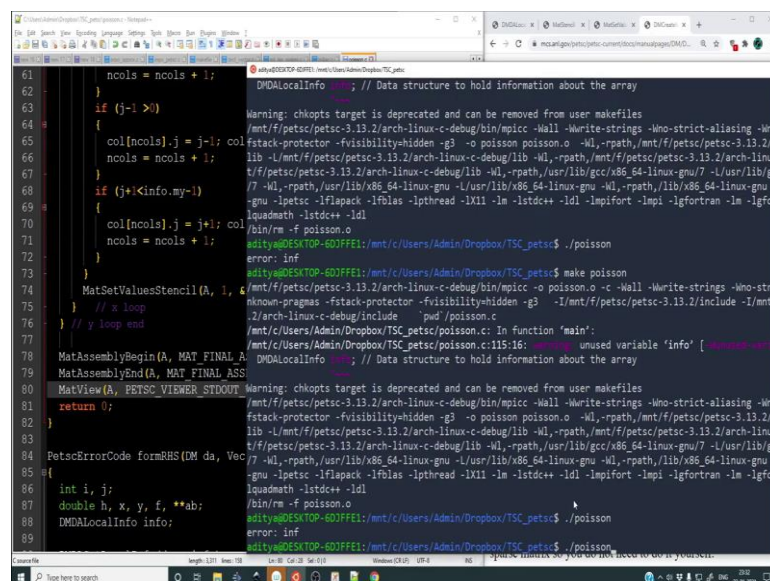
n the stack are not available,
of the start of the function
6 /mnt/f/petsc/petsc-3.13.2/src/sys/memory/mtr.c
9 /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/r/vecro
101 /mnt/f/petsc/petsc-3.13.2/src/dm/impls/da/da/getarray.
----- Error Message -----
ts/documentation/faq.html for trouble shooting.
Jun 02, 2020
bug named DESKTOP-60JFFE1 by aditya Fri Jan 22 23:29:23
c --with-cxx-g++ --with-fc-gfortran --download-mpich --d
line 0 in unknown file
50152859) - process 0
t exitcode=50152859
descriptor
ov/TSC_petsc\$ make poisson
in/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-stri
hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/
/c
t: In function 'u_exact':
:22:3: error: expected expression before '/' token
*/
c: In function 'main':
:115:16: warning: unused variable 'info' [-Wunused-vari
d information about the array
:259: recipe for target 'poisson.o' failed
ov/TSC_petsc\$

Everything seems to run but still we have an error, so in fact, because we have some kind of error. Let us quickly try to debug it. So, for that, we are going to print out the where is the we can print the different things.

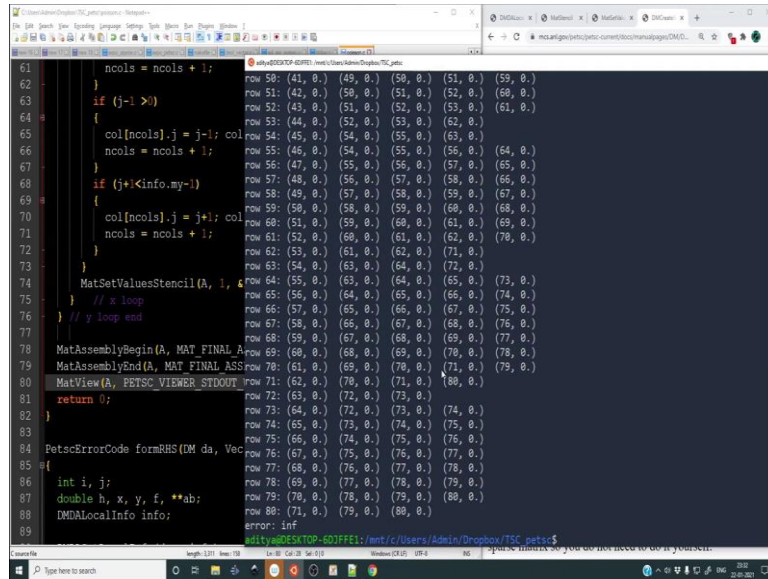
(Refer Slide Time: 62:11)



(Refer Slide Time: 62:19)



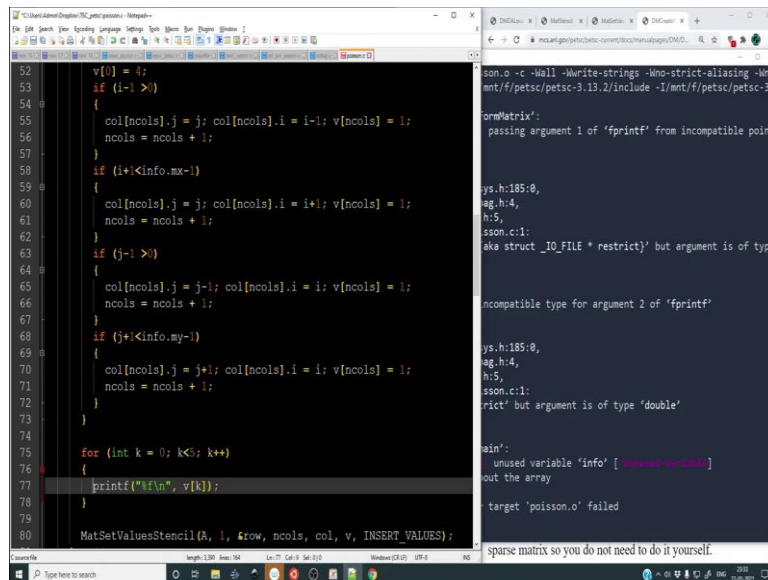
(Refer Slide Time: 62:27)



```
61     ncols = ncols + 1;
62   }
63   if (j-1 > 0)
64   {
65     col[ncols].j = j-1; col
66     ncols = ncols + 1;
67   }
68   if (j+1 < info.mx-1)
69   {
70     col[ncols].j = j+1; col
71     ncols = ncols + 1;
72   }
73   MatSetValuesStencil(A, 1, c
74   } // x loop
75   // y loop end
76   row 50: (41, 0.) (48, 0.) (50, 0.) (51, 0.) (59, 0.)
77   row 51: (42, 0.) (50, 0.) (51, 0.) (52, 0.) (58, 0.)
78   row 52: (43, 0.) (51, 0.) (52, 0.) (53, 0.) (61, 0.)
79   row 53: (44, 0.) (52, 0.) (53, 0.) (62, 0.)
80   row 54: (45, 0.) (54, 0.) (55, 0.) (63, 0.)
81   row 55: (46, 0.) (54, 0.) (55, 0.) (56, 0.) (64, 0.)
82   row 56: (47, 0.) (55, 0.) (56, 0.) (57, 0.) (65, 0.)
83   row 57: (48, 0.) (56, 0.) (57, 0.) (58, 0.) (66, 0.)
84   row 58: (49, 0.) (57, 0.) (58, 0.) (59, 0.) (67, 0.)
85   row 59: (50, 0.) (58, 0.) (59, 0.) (60, 0.) (68, 0.)
86   row 60: (51, 0.) (59, 0.) (60, 0.) (61, 0.) (69, 0.)
87   row 61: (52, 0.) (60, 0.) (61, 0.) (62, 0.) (70, 0.)
88   row 62: (53, 0.) (61, 0.) (62, 0.) (71, 0.)
89   row 63: (54, 0.) (63, 0.) (64, 0.) (72, 0.)
90   row 64: (55, 0.) (63, 0.) (64, 0.) (65, 0.) (73, 0.)
91   row 65: (56, 0.) (64, 0.) (65, 0.) (66, 0.) (74, 0.)
92   row 66: (57, 0.) (65, 0.) (66, 0.) (67, 0.) (75, 0.)
93   row 67: (58, 0.) (66, 0.) (67, 0.) (68, 0.) (76, 0.)
94   row 68: (59, 0.) (67, 0.) (68, 0.) (69, 0.) (77, 0.)
95   row 69: (60, 0.) (68, 0.) (69, 0.) (70, 0.) (78, 0.)
96   row 70: (61, 0.) (69, 0.) (70, 0.) (71, 0.) (79, 0.)
97   row 71: (62, 0.) (70, 0.) (71, 0.) (80, 0.)
98   row 72: (63, 0.) (72, 0.) (73, 0.)
99   row 73: (64, 0.) (72, 0.) (73, 0.) (74, 0.)
100  row 74: (65, 0.) (73, 0.) (74, 0.) (75, 0.)
101  row 75: (66, 0.) (74, 0.) (75, 0.) (76, 0.)
102  row 76: (67, 0.) (75, 0.) (76, 0.) (77, 0.)
103  row 77: (68, 0.) (76, 0.) (77, 0.) (78, 0.)
104  row 78: (69, 0.) (77, 0.) (78, 0.) (79, 0.)
105  row 79: (70, 0.) (78, 0.) (79, 0.) (80, 0.)
106  row 80: (71, 0.) (79, 0.) (80, 0.)
107  error: inf
108  PetscErrorCode formRHS(DM da, Vec
109  {
110  int i, j;
111  double h, x, y, f, **ab;
112  DMLocalInfo info;
113  PetscErrorCode formRHS(DM da, Vec
```

Well, when we view the matrix, everything appears to be 0. So, why is that? Why is everything 0? So, let us make a small loop over here to print out the values of b that are being inserted.

(Refer Slide Time: 62:49)



```
52     v[i] = 4;
53     if (i-1 > 0)
54     {
55       col[ncols].j = j; col[ncols].i = i-1; v[ncols] = 1;
56       ncols = ncols + 1;
57     }
58     if (i+1 < info.mx-1)
59     {
60       col[ncols].j = j; col[ncols].i = i+1; v[ncols] = 1;
61       ncols = ncols + 1;
62     }
63     if (j-1 > 0)
64     {
65       col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
66       ncols = ncols + 1;
67     }
68     if (j+1 < info.mx-1)
69     {
70       col[ncols].j = j+1; col[ncols].i = i; v[ncols] = 1;
71       ncols = ncols + 1;
72     }
73   }
74   for (int k = 0; k < 5; k++)
75   {
76     printf("%f \n", v[k]);
77   }
78   MatSetValuesStencil(A, 1, &row, ncols, col, v, INSERT_VALUES);
```

For int k = 0, k < 5, k ++, fprintf % f \ n, v k. So, this should help us set things straight why that error is. So, let me make this not fprintf, this has to be simply printf.

(Refer Slide Time: 63:43)

```
121 DMALocalInfo info; // Data structure to hold information about the
    array
122
123 PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
124
125 DMACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE,
    DMDA_STENCIL_STAR, 9, 9, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL,
    NULL, &da);
126
127 DMSetFromOptions(da);
128 DMSetUp(da);
129 DMCreateMatrix(da, &A);
130 MatSetFromOptions(A);
131
132 DMCreateGlobalVector(da, &b);
133 VecDuplicate(b, &u);
134 VecDuplicate(b, &uexact);
135
136 // form Exact solution
137 u_exact(da, uexact);
138 // form the matrix A
139 formMatrix(da, A);
140 // Form the RHS bar
141 formRHS(da, b);
142
143 KSPCreate(PETSC_COMM_WORLD, &ksp);
144 KSPSetOperators(ksp, A, A);
145 KSPSetFromOptions(ksp);
146 KSPSolve(ksp, b, u);
```

```
ys.h:185:0:
  eg.h:4,
  h:5,
  sson.c:1:
    rict' but argument is of type 'double'
  ain':
    unused variable 'info' [ -Wunused-variable]
  out the array
  'target 'poisson.o' failed
  make poisson
  son.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno
  mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.
  ain':
    unused variable 'info' [ -Wunused-variable]
  out the array
  user makefiles
  Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas
  -Ml, -rpath, /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debu
  ath, /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debu/lib -U
  lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
  x-gnu -Ml, -rpath, /lib/x86_64-linux-gnu -L/lib/x86_64-lin
  d -lmpifort -lmpi -lgfortran -lm -lgfortran -lm -lgcc_s
  /poisson
  sparse matrix so you do not need to do it yourself.
```

I think I have completely lost it, because I had commented them out.

(Refer Slide Time: 63:47)

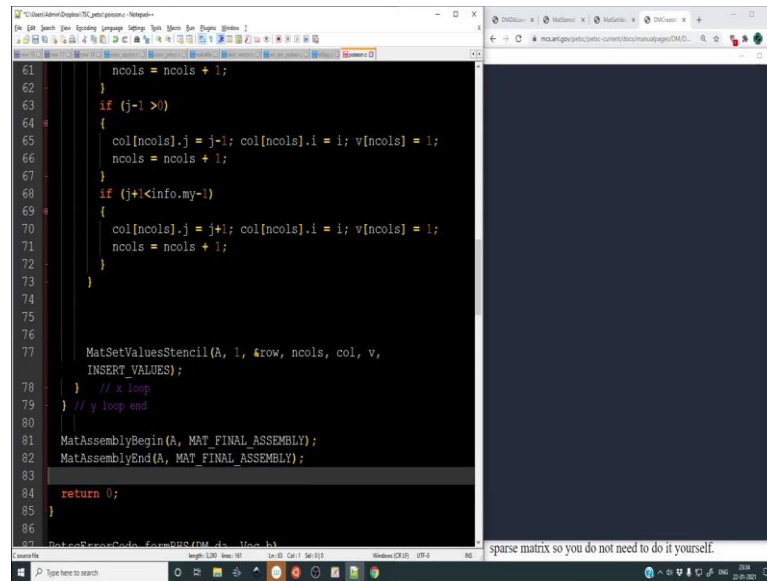
```
121 DMALocalInfo info; // Data structure to hold information about the
    array
122
123 PetscInitialize(&argc, &argv, NULL, "Solve Poisson equation in 2D");
124
125 DMACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_NONE,
    DMDA_STENCIL_STAR, 9, 9, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL,
    NULL, &da);
126
127 DMSetFromOptions(da);
128 DMSetUp(da);
129 DMCreateMatrix(da, &A);
130 MatSetFromOptions(A);
131
132 DMCreateGlobalVector(da, &b);
133 VecDuplicate(b, &u);
134 VecDuplicate(b, &uexact);
135
136 // form Exact solution
137 u_exact(da, uexact);
138 // form the matrix A
139 formMatrix(da, A);
140 // Form the RHS bar
141 formRHS(da, b);
142
143 KSPCreate(PETSC_COMM_WORLD, &ksp);
144 KSPSetOperators(ksp, A, A);
145 KSPSetFromOptions(ksp);
146 KSPSolve(ksp, b, u);
```

```
row 50: (41, 1.) (49, 1.) (50, 4.) (51, 1.) (59, 1.)
row 51: (42, 1.) (50, 1.) (51, 4.) (52, 1.) (60, 1.)
row 52: (45, 1.) (51, 1.) (52, 4.) (53, 0.) (61, 1.)
row 53: (44, 0.) (52, 0.) (53, 1.) (62, 0.)
row 54: (45, 0.) (54, 1.) (55, 0.) (63, 0.)
row 55: (46, 1.) (54, 0.) (55, 4.) (56, 1.) (64, 1.)
row 56: (47, 1.) (55, 1.) (56, 4.) (57, 1.) (65, 1.)
row 57: (48, 1.) (56, 1.) (57, 4.) (58, 1.) (66, 1.)
row 58: (49, 1.) (57, 1.) (58, 4.) (59, 1.) (67, 1.)
row 59: (50, 1.) (58, 1.) (59, 4.) (60, 1.) (68, 1.)
row 60: (51, 1.) (59, 1.) (60, 4.) (61, 1.) (69, 1.)
row 61: (52, 1.) (60, 1.) (61, 4.) (62, 0.) (70, 1.)
row 62: (53, 0.) (61, 0.) (62, 1.) (71, 0.)
row 63: (54, 0.) (63, 1.) (64, 0.) (72, 0.)
row 64: (55, 1.) (63, 0.) (64, 4.) (65, 1.) (73, 0.)
row 65: (56, 1.) (64, 1.) (65, 4.) (66, 1.) (74, 0.)
row 66: (57, 1.) (65, 1.) (66, 4.) (67, 1.) (75, 0.)
row 67: (58, 1.) (66, 1.) (67, 4.) (68, 1.) (76, 0.)
row 68: (59, 1.) (67, 1.) (68, 4.) (69, 1.) (77, 0.)
row 69: (60, 1.) (68, 1.) (69, 4.) (70, 1.) (78, 0.)
row 70: (61, 1.) (69, 1.) (70, 4.) (71, 0.) (79, 0.)
row 71: (62, 0.) (70, 0.) (71, 1.) (80, 0.)
row 72: (63, 0.) (72, 1.) (73, 0.)
row 73: (64, 0.) (73, 0.) (74, 1.) (74, 0.)
row 74: (65, 0.) (75, 0.) (76, 1.) (75, 0.)
row 75: (66, 0.) (74, 0.) (75, 1.) (76, 0.)
row 76: (67, 0.) (75, 0.) (76, 1.) (77, 0.)
row 77: (68, 0.) (76, 0.) (77, 1.) (78, 0.)
row 78: (69, 0.) (77, 0.) (78, 1.) (79, 0.)
row 79: (70, 0.) (78, 0.) (79, 1.) (80, 0.)
row 80: (71, 0.) (79, 0.) (80, 1.)
KSPSetFromOptions error: 5.6155592e-02
KSPSolve(ksp, b, u);
```

```
sparse matrix so you do not need to do it yourself.
```

Well, I hope it runs now. Yes, and the error is $5.6 \cdot 10^{-2}$. Well, there was nothing wrong. It is just that I had commented it out for debugging something else, and forgot to uncomment it back, those kind of thing happen especially when you are doing things live alright. Let me remove the small print statement that we have done great. And we do not need to print the matrix as well.

(Refer Slide Time: 64:25)

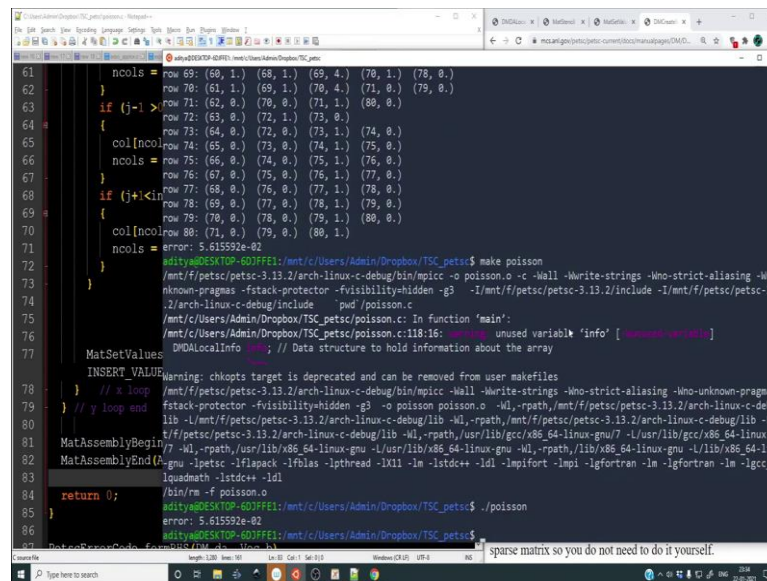


```
61     ncols = ncols + 1;
62     }
63     if (j-1 > 0)
64     {
65         col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
66         ncols = ncols + 1;
67     }
68     if (j+1 < info.my-1)
69     {
70         col[ncols].j = j+1; col[ncols].i = i; v[ncols] = 1;
71         ncols = ncols + 1;
72     }
73     }
74
75     MatSetValuesStencil(A, 1, 4*row, ncols, col, v,
76     INSERT_VALUES);
77     } // x loop
78 } // y loop end
79
80 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
81 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
82
83 return 0;
84 }
85 }
86 }
87 }
```

sparse matrix so you do not need to do it yourself.

So, yeah that takes care of that. Let us make it.

(Refer Slide Time: 64:31)



```
61     ncols = row 69: (69, 1.) (68, 1.) (69, 4.) (70, 1.) (78, 0.)
62     }
63     row 70: (61, 1.) (69, 1.) (70, 4.) (71, 0.) (79, 0.)
64     if (j-1 > row 71: (62, 0.) (70, 0.) (71, 1.) (80, 0.)
65     {
66     row 72: (63, 0.) (72, 1.) (73, 0.)
67     col[ncol row 73: (64, 0.) (72, 0.) (73, 1.) (74, 0.)
68     }
69     ncols = row 74: (65, 0.) (73, 0.) (74, 1.) (75, 0.)
70     row 75: (66, 0.) (74, 0.) (75, 1.) (76, 0.)
71     row 76: (67, 0.) (75, 0.) (76, 1.) (77, 0.)
72     if (j+1 < row 77: (68, 0.) (76, 0.) (77, 1.) (78, 0.)
73     row 78: (69, 0.) (77, 0.) (78, 1.) (79, 0.)
74     {
75     col[ncol row 79: (70, 0.) (78, 0.) (79, 1.) (80, 0.)
76     row 80: (71, 0.) (79, 0.) (80, 1.)
77     ncols = error: 5.615592e-02
78     }
79     }
80     MatSetValues warning: ckopts target is deprecated and can be removed from user makefiles
81     INSERT_VALUES warning: ckopts target is deprecated and can be removed from user makefiles
82     } // x loop /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas
83     } // y loop end fstack-protector -fvisibility-hidden -g3 -o poisson.o -M-,rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib
84 MatAssemblyBegin(/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -M-,rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -M-,rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
85 MatAssemblyEnd(/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -M-,rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
86 return 0; /bin/rm -f poisson.o
87 }
88 }
89 }
```

error: 5.615592e-02

error: 5.615592e-02

sparse matrix so you do not need to do it yourself.

And here it is this.

(Refer Slide Time: 64:41)

```
129 DMCreateGlobalVector(da, &b);
130 VecDuplicate(b, &u);
131 VecDuplicate(b, &uexact);
132
133 // form Exact solution
134 u_exact(da, uexact);
135 // form the matrix A
136 formMatrix(da, A);
137 // form the RHS b
138 formRHS(da, b);
139
140 KSPCreate(PETSC_COMM_WORLD, &ksp);
141 KSPSetOperators(ksp, A, A);
142 KSPSetFromOptions(ksp);
143 KSPSolve(ksp, b, u);
144
145 VecXPY(u, -1, uexact); // easy
146 VecNorm(u, NORM_INFINITY, &err);
147
148 PetscPrintf(PETSC_COMM_WORLD, "grid size: %d error: %e\n", err);
149
150 VecDestroy(&u);
151 VecDestroy(&uexact);
152 VecDestroy(&b);
153 MatDestroy(&A);
154 KSPDestroy(&ksp);
155 DMDestroy(&da);
156
157 return PetscFinalize();
```

In fact, in the main, we can tell the grid size percentage d and corresponding to that grid size what the error is. And for fetching the grid size, what we need to do is `DMDAGetLocalInfo`, then `da` into `info`. And finally, we have to print `info dot m x` that is the size of the grid nothing else right. So, and it is equal in both directions, so does not matter. So, let me recompile.

(Refer Slide Time: 65:23)

```
129 DMCreateGlobalVector(da, &b);
130 VecDuplicate(b, &u);
131 VecDuplicate(b, &uexact);
132
133 // form Exact solution
134 u_exact(da, uexact);
135 // form the matrix A
136 formMatrix(da, A);
137 // form the RHS b
138 formRHS(da, b);
139
140 KSPCreate(PETSC_COMM_WORLD, &ksp);
141 KSPSetOperators(ksp, A, A);
142 KSPSetFromOptions(ksp);
143 KSPSolve(ksp, b, u);
144
145 VecXPY(u, -1, uexact); // easy
146 VecNorm(u, NORM_INFINITY, &err);
147
148 PetscPrintf(PETSC_COMM_WORLD, "grid size: %d error: %e\n", err);
149
150 VecDestroy(&u);
151 VecDestroy(&uexact);
152 VecDestroy(&b);
153 MatDestroy(&A);
154 KSPDestroy(&ksp);
155 DMDestroy(&da);
156
157 return PetscFinalize();
```

And so grid size is 9. So, now, what we can do is we can pass we can refine the grid. So, minus `da refine`. And what `refine` does is it doubles the number of grid points ok. So,

refine simply doubles the number of grid points. So, if I do 2, it runs the code on 33×33 . What does refine 4 do? To 129. So, grid size is 9 yeah. So, if refine is 1, it will do it on a grid size of 17 and so on. So, it is 9, 17, 33 and so on. So, this is how we can increase the refinement.

So, let me refine something. Why is the error not reducing a lot? Why is the error stagnating? So, let us see whether we have so most likely its some error in writing these functions because we do not seem to have done anything wrong, no, that is correct.

(Refer Slide Time: 67:25)

```

39 for (i = info.xs; i < (info.xs + info.xm); i++)
40 {
41     row.j = j; row.i = i;
42     col[0].j = j; col[0].i = i;
43
44     ncols = 1;
45
46     if (i == 0 || i == info.mx - 1 || j == 0 || j == info.my - 1)
47     {
48         v[0] = 1.0;
49     }
50     else
51     {
52         v[0] = 4;
53         if (i-1 > 0)
54         {
55             col[ncols].j = j; col[ncols].i = i-1; v[ncols] = 1;
56             ncols = ncols + 1;
57         }
58         if (i+1 < info.mx-1)
59         {
60             col[ncols].j = j; col[ncols].i = i+1; v[ncols] = 1;
61             ncols = ncols + 1;
62         }
63         if (j-1 > 0)
64         {
65             col[ncols].j = j-1; col[ncols].i = i; v[ncols] = 1;
66             ncols = ncols + 1;
67     }

```

```

lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
x-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-lin
dl -lmpifort -lmpi -lgfortran -lm -lgfortran -lm -lgcc_s
/poisson
/poisson -da_refine 2
/poisson -da_refine 4
/poisson -da_refine 1
/poisson -da_refine 5
make poisson
son.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno
mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.
user makefiles
Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas
-Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debu
ath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/
lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
x-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-lin
dl -lmpifort -lmpi -lgfortran -lm -lgfortran -lm -lgcc_s
/poisson -da_refine 5
/poisson -da_refine 5
sparse matrix so you do not need to do it yourself.

```

So, uexact these all should have been -1. Well, you have seen me make some mistakes, but I hope that does not distract you from the overall structure of the program these are all were minus 1s. And it pays that is why it is always good to have a known solution. So, that you know if something is going wrong or not, yeah. So, let me recompile.

(Refer Slide Time: 67:53)

```
91 DMDALocalInfo info;
92 /bin/rm -f poisson.o
93 DMDAGetLocalInfo aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 5
94 h = 1.0/(info.mx-grid size: 257 error: 6.501178e+00
95 DMDAVecGetArray(/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-
96 for (j = info.ys; known-pragmas -fstack-protector -fvisibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.
97 2/arch-linux-c-debug/include -pwl /poisson.c
98 y = j*h; Warning: chkopts target is deprecated and can be removed from user makefiles
99 for (i = info.x;/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas
100 { fstack-protector -fvisibility-hidden -g3 -o poisson poisson.o -Ml, -rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug
101 lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Ml, -rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/
102 t/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Ml, -rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-g
103 f = 2*(1-c*x)/7 -Ml, -rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Ml, -rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-lin
104 ab[j][i] = f*
105 }
106 /bin/rm -f poisson.o
107 DMDAVecRestoreArr aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 1
108 grid size: 17 error: 9.736422e-03
109 return 0; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 2
110 grid size: 33 error: 2.438715e-03
111 aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 3
112 grid size: 65 error: 6.102748e-04
113 int main(int argc, aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 4
114 =() grid size: 129 error: 1.525694e-04
115 DM da; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 7
116 Mat A; ^C
117 Vec b, u, uexact; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 6
118 KSP ksp; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ^C
119 double err; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 5
120 DMDALocalInfo info; grid size: 257 error: 3.814691e-05
121 array; aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 2 -mat_view draw -draw_pause 10
```

This time we should have it correct, yes. So, as we increase the refinement, the error keeps on reducing, maybe it is too large of a grid. Let us see. Let us wait for a while and maybe it is too large let me make it 6. This should not take too much time on my computer. But yeah you can see that the error is reducing. The point I want to make is do not make silly mistakes that is all I can say.

So, in this particular lecture, we have seen how to solve a Poisson equation using this DMDA a structured grid. And once this runs, I can we can visualize the matrix and all that as well. Well, let us just do it for 5 and yeah. So, for 5, it is this. And let me only refine it twice. And let me show you the matrix. So, minus mat view draw minus draw pause 10. Before that, I have to start x ming which is the x 11 forwarder for communicating for creating graphics from this virtual sort of machine, yeah.

(Refer Slide Time: 69:39)

```

inf:grid size: 257 error: 6.581178e+08
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ make poisson
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-
info: /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o poisson.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-
unknown-pragmas -fstack-protector -fvisibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.
.mt/arch-linux-c-debug/include -pwl/poisson.c
Warning: chlopts target is deprecated and can be removed from user makefiles
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 1
grid size: 17 error: 9.736422e-03
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 2
grid size: 33 error: 2.438715e-03
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 3
grid size: 65 error: 6.102748e-04
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 4
grid size: 129 error: 1.525594e-04
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 7
grid size: 257 error: 3.814691e-05
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 6
grid size: 513 error: 9.536743e-06
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 5
grid size: 257 error: 3.814691e-05
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 2 -mat_view draw -draw_pause 10
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./poisson -da_refine 1 -mat_view draw -draw_pause 10
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$

```

So, this is how the Penta diagonal matrix looks like. Let me make it only 1, maybe it is better. Yeah, this is how the Penta diagonal matrix looks like. Well, unfortunately when I expand this, it goes away anyway. So, you can have a look at that. And what else can I show? I can yeah. So, you can output the file. I have shown you how to output a file using `fprintf`, and you can plot it using Python and all.

But anyway the key point that I want to show is how to make the RHS, how to make the LHS. So, now, you can try to solve various kinds of partial differential equations using the DMDA construct. So, finite difference methods you can easily solve. So, I am not going to take any much of your time. We have begun we have begun making some silly mistakes, but anyway. I will see you again next time, we are going to do some non-linear equations.

Until then it is good bye. Have a good day.