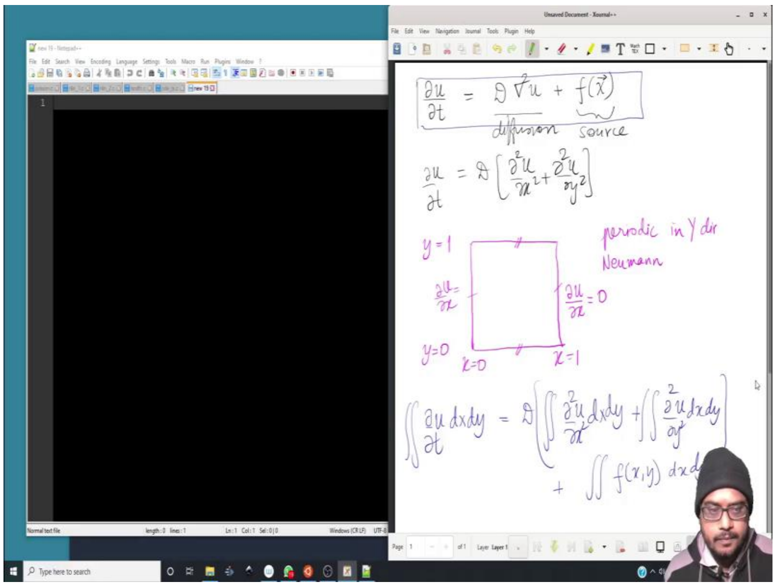
# Tools in Scientific Computing

**Prof. Aditya Bandopadhyay**
**Department of Mechanical Engineering**
**Indian Institute of Technology, Kharagpur**

## Lecture - 37
## Heat transport using PETSc

Hi everyone. In this particular lecture, we are going to look at a Two-Dimensional Heat Equation. So, let us consider a non-steady heat conduction problem with source.

(Refer Slide Time: 00:46)


$\boxed{\frac{\partial u}{\partial t} = \underbrace{D\nabla^2 u}_{\text{diffusion}} + \underbrace{f(\vec{x})}_{\text{source}}}$
$\frac{\partial u}{\partial t} = D\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right]$
periodic in Y dir
Neumann
y = 1
y = 0
x = 0
x = 1
$\frac{\partial u}{\partial x} =$
$\frac{\partial u}{\partial x} = 0$
$\iint \frac{\partial u}{\partial t}\,dx\,dy = D\left[\iint \frac{\partial^2 u}{\partial x^2}dx\,dy + \iint \frac{\partial^2 u}{\partial y^2}dx\,dy\right] + \iint f(x,y)\,dx\,d$

So, let the variable under question be u, just to not confuse it with small t for time. So, the governing differential equation, it is a partial differential equation which says del u del t is some diffusivity could be thermal conductivity, it could be molecular diffusivity is equal to its in 2D. So, the application of u plus f of the space coordinate. It could be a function of time, but let us assume that it is only a function of the space coordinate.

Well, now this particular term accounts for the time rate of change. This accounts for the diffusion and this is the source. And we have learnt in elementary heat transfer course that depending on the geometry of the problem, the Laplacian could be written in general as a single coordinate for; for example, for a fin.

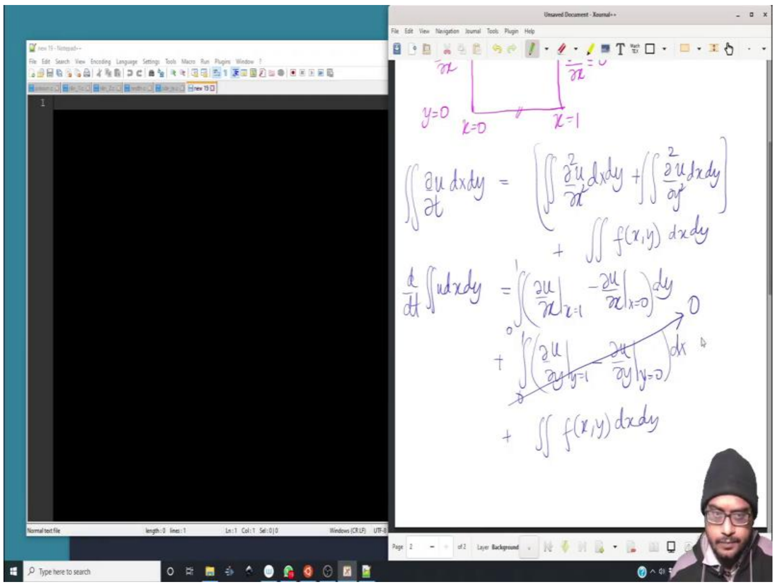
When you know that heat transfer is going to be predominant along one direction, but this is the general form and in two-dimension the Laplacian is going to be del 2 u del x 2 plus del 2 u del y 2, alright. So, under what conditions is this equation consistent? So, let us first define what the domain will be over with the solution will be valid.

So, consider a domain like this, this is x equal to 0, x equal to 1, this is y equal to 0, and this is y equal to 1. So, let these two boundaries be periodic. So, it is periodic in the in Y direction. And let these two boundaries be Neumann boundary condition. So, let this particular boundary be del u by del x is equal to 0, that is a no flux boundary.

We have to assign some boundary condition over here, maybe in terms of some function of y. So, let us see what whether there is a bound on that function of y or not, right. So, how do we find out that consistency rule? So, let us integrate this equation over space. So, when you integrate this equation.

So, you have integral del u del t dx dy is equal to d integral del 2 u del x 2, dx dy plus integral del 2 u del y 2 dx dy plus integral. So, double integral because it is over an area plus double integral f of x comma y dx dy. So, now because the area over which were integrating is not changing, so we can take this del del t out and make it d dt, so it is d dt of this integral.

(Refer Slide Time: 04:22)

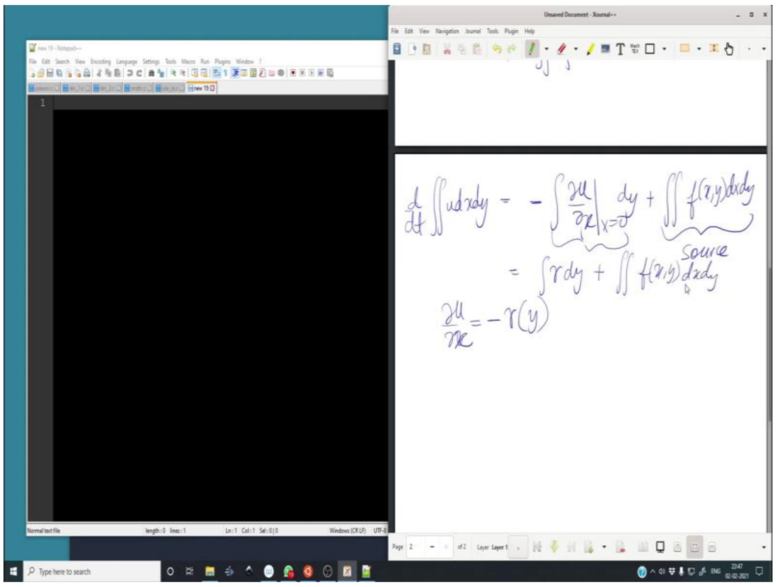


$$\iint \frac{\partial u}{\partial t}\,dx\,dy = \left[\iint \frac{\partial^2 u}{\partial x^2}\,dx\,dy + \iint \frac{\partial^2 u}{\partial y^2}\,dx\,dy\right] + \iint f(x,y)\,dx\,dy$$

$$\frac{d}{dt}\iint u\,dx\,dy = \int_0^1 \left(\frac{\partial u}{\partial x}\Big|_{x=1} - \frac{\partial u}{\partial x}\Big|_{x=0}\right)dy + \cancel{\int_0^1 \left(\frac{\partial u}{\partial y}\Big|_{y=1} - \frac{\partial u}{\partial y}\Big|_{y=0}\right)dx} \rightarrow 0 + \iint f(x,y)\,dx\,dy$$

So, it is the integral of u over the entire domain is equal to. So, now, what is this first term going to be? So, well let us assume that this diffusivity is one for simplicity. So, it is going to be del u del x, x equal to 1 minus del u del x, x equal to 0 times integral dy plus the second term will be integral del u del y, y equal to 1 minus del u del y at y equal to 0, dx. So, all these integrals are from 0 to 1 plus double integral of f of x, y dx dy. Now, what about this? Because it is periodic in the Y direction, so del u del y at y equal to 1 is identical to del u del y at y equal to 0. And hence this particular term it vanishes.

(Refer Slide Time: 05:32)



$$\frac{d}{dt}\iint u\,dx\,dy = -\underbrace{\int \left.\frac{\partial u}{\partial x}\right|_{x=0} dy} + \underbrace{\iint f(x,y)\,dx\,dy}_{\text{source}}$$

$$= \int \gamma\,dy + \iint f(x,y)\,dx\,dy$$

$$\frac{\partial u}{\partial x} = -\gamma(y)$$

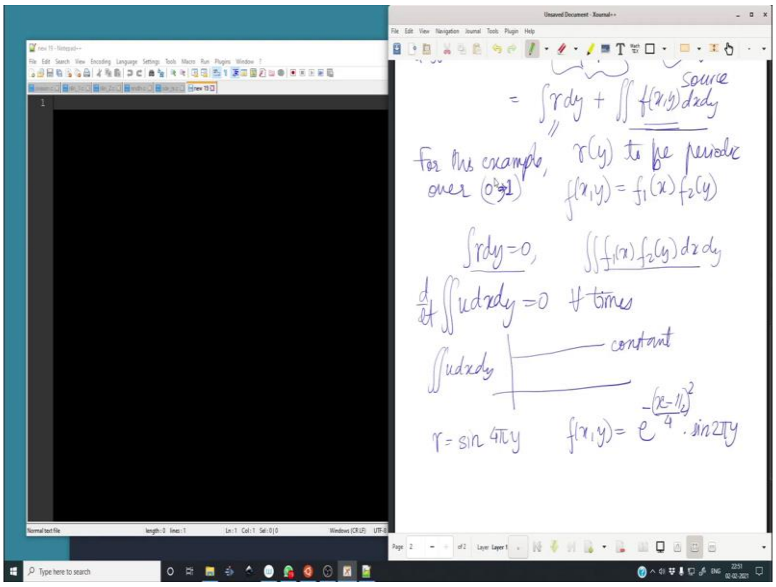
So, now we have d dt of this integral u dx dy is equal to; so, now, del u del x at x equal to 1, we have said that its going to be a Neumann boundary condition, it is a no flux boundary condition rather. So, we have minus integral of del u del y at x equal to 0 dy plus integral of f of x comma y dx dy.

So, now because it is a problem in which we have no flux boundary periodic boundaries. There has to be this particular quantity which is going to be conserved, so d dt of this conserved quantity has to be equal to 0. So, that particular quantity is like the total energy of the system or the total mass of the system, whatever you want to give it as. So, whatever is being generated has to be sort of be balanced by the time rate of change.

The time rate of change is, so whatever is being whatever the source is has to be sort of absorbed by this boundary, right. So, let us say that del u del y is equal to minus gamma of y. So, del u del y at x equal to 0 I think, so it is del u del x at x equal to 0 (Refer Time: 07:17). So, let del u del x at x equal to 0 be equal to minus of gamma, minus is just to absorb this sign and so, this remains the same.

So, this essentially becomes integral gamma dy plus integral of f x comma y dxdy. And so, if everything has to be consistent; well, actually this need not be 0, you can have more source, less absorption, so this quantity can change. But we can choose the form of gamma and the form of f whatever we want in this particular example.

(Refer Slide Time: 08:05)



$$= \int \gamma \, dy + \iint \underbrace{f(x,y)}_{} \overset{\text{source}}{dx\,dy}$$

For this example, $\gamma(y)$ to be periodic over $(0 \to 1)$

$$f(x,y) = f_1(x) f_2(y)$$

$$\int \gamma \, dy = 0, \quad \iint f_1(x) f_2(y)\, dx\, dy$$

$$\frac{d}{dt} \iint u \, dx\, dy = 0 \quad \forall \text{ times}$$

$\iint u \, dx\, dy$ constant

$$\gamma = \sin 4\pi y \qquad f(x,y) = e^{-\frac{(x-1/2)^2}{4}} \cdot \sin 2\pi y$$

So, for this example we can choose gamma of y to be a periodic function over 0 to 1 and we can choose f of x, y to be some separable form like f 1 of x, f 2 of y. And if f 2 of y is periodic over 0 to 1 then automatically this integral will also become, so this integral will become 0. So, if gamma is periodic over 0 to 1 then integral gamma dy will be 0, and if f of x, y has a separable form such that it is this then integral f 1 x, f 2 y, dxdy will also vanish.

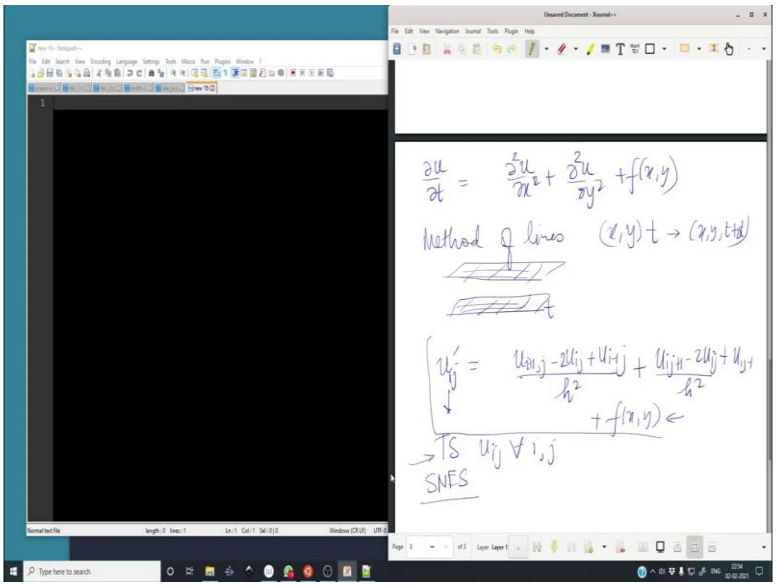
So, in order to construct these functions, what are the ramifications if we construct the functions like this? The ramification is this is going to be 0, this is going to be 0. So, d dt of this integral of u dxdy at all times will be equal to 0. Meaning if we were to solve the system and we were to monitor this particular integral it would be some constant, ok and that is what I mean it should come out to be it has to be equal to a constant, alright.

So, what do we have so far? We have we need to choose this these two particular forms. So, fine, no problem. So, let us choose gamma to be sin of 4 pi y is periodic over 0 to 1 and let f of x, y. So, let it be e to the power minus x minus half whole square divided by say 4 times sin 2 xy; so, 2 pi y not 2 xy. So, this integral will also be 0 because if we choose this form and similarly this integral will be also 0.

So, there you have we have constructed a synthetic problem which is consistent, we cannot just choose any random boundary conditions because if you choose any random boundary conditions there is no guarantee where the solution will exist or not.

There has to be this kind of a condition. And the beauty of this form that we have chosen synthetically is that the quantity integral of u over the area has to be conserved, ok. So, let us try to solve this numerically. Let us try to discretize it first.

(Refer Slide Time: 11:12)



$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x,y)$$

Method of lines $(x,y)t \rightarrow (x,y,t+\delta t)$

$$u_{ij}' = \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h^2} + f(x,y)$$

TS $u_{ij} \; \forall \; i,j$

SNES

So, in the discrimination what do we have? So, del u del t, del 2 u del x 2, del 2 u del y 2 plus f of x, y. So, over here we can use the method of lines we have sort of mapping from t x, some solution at x comma and all the x, y that is at certain time and you move to a solution of all x, y, z p plus delta t you sort of move from one time level to another time level. So, you can have all the x, y solutions over here.

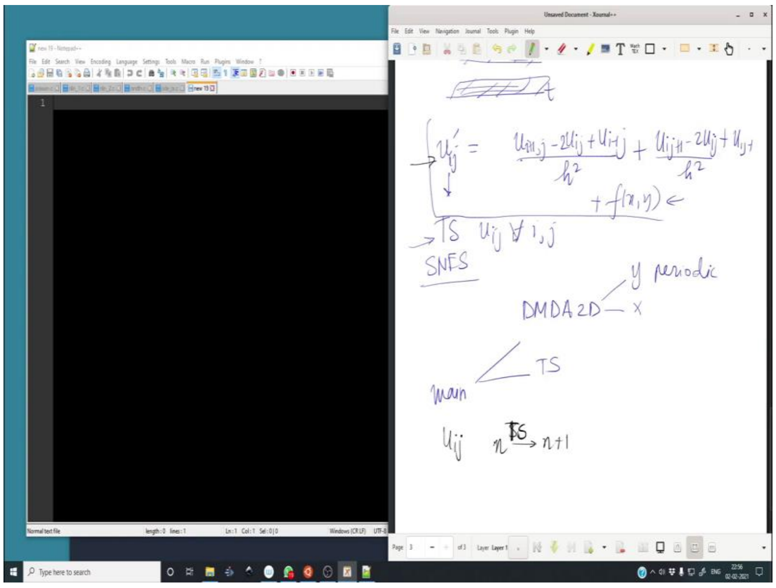
So, let us write down the discretization of this. So, it is u prime, so du dt is u prime and this is equal to the discrete form of u at that particular time. So, its u at i plus 1 j minus u ij minus 2 u ij plus u i minus 1 j upon h square plus u ij plus 1 minus 2 u ij plus u ij minus 1 upon h square plus f of x, y. So, it is like you are forming a differential equation for u ij, ok. And through this we can then form what we say as a differential equation for u ij, solve this non-linear equation.

So, in general because of this nonlinearity this, this will form a non-linear. So, basically you have to time step for u ij for all i's and j's. So, it is time stepping for u ij for all i's and j's, and we can have various kinds of methods in order to perform the time stepping and we have seen already the time stepping in, but see is not that difficult at all.

But look because it is going to form sort of a non-linear system because of these functions. So, it is like having both time stepping and an SNES object. It is going to contain both a time stepping scheme and an SNES, ok. So, let us make use of all the information we have learnt so far.

So, essentially we have learnt how to create a grid, we have seen how to create the time stepping routine, we have also seen how to sort of make. Not; in this case we will not need to make an SNES object separately, but we will sort of make use of that non-linear solver also. Well, it may not be so obvious how to go about doing that, but as we do that we will see how it looks. But so, the main sort of flowchart of the particular problem would be the following.

(Refer Slide Time: 14:44)



$$u_{ij}' = \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{ij+1} - 2u_{ij} + u_{ij-1}}{h^2} + f(x,y)$$

TS $u_{ij}\ \forall\ i,j$

SNES

$$\begin{tikzcd} & y \text{ periodic} \\ \text{DMDA 2D} \arrow[ur, dash] \arrow[r, dash] & x \end{tikzcd}$$

$$\begin{tikzcd} & \text{TS} \\ \text{main} \arrow[ur, dash] & \end{tikzcd}$$

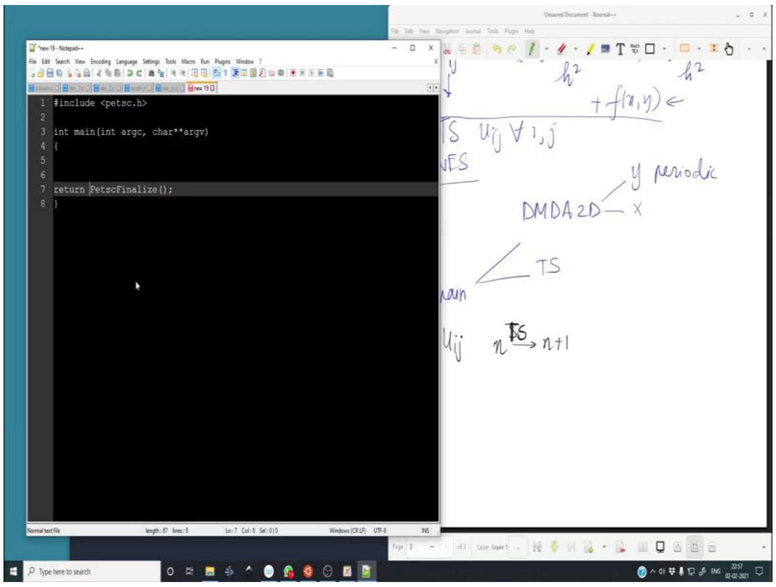
$u_{ij} \quad n \xrightarrow{\text{TS}} n+1$

So, what we will need is we will be first have to make a main program that is always there. From this main program we must first set up a DMDA 2D, that is the grid over which everything has to be defined. You have to make the Y direction periodic. We have to sort of discretize the X direction to account for the two Neumann boundary conditions, one boundary condition particular being no flux boundary condition.

So, after setting this we must set the time stepping object. Inside the time stepping we can define the Jacobian. So, how do you go about finding the Jacobian? It is like u prime equal to some function of this. So, what are the variables over here? And if you are attentive you might easily guess that the variables are all the u ij. So, once you solve this you sort of achieve a time stepping to go from the nth level to n plus 1th level, ok. So, this is done by time stepping.

But the time stepping will form a non-linear algebraic equation, ok. It will form a non-linear algebraic equation which you then need to solve and to account for this you need to somehow tell the time stepping, the time stepper that well not the time stepper, but the DMDA that you are going to also set a Jacobian in this function. So, you have to basically link that particular Jacobian to this particular set of objects, ok. So, let us take things one by one.

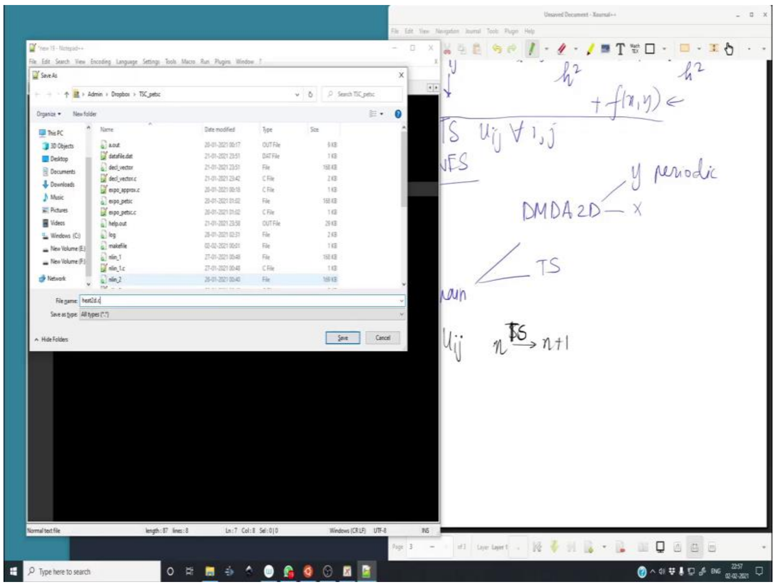
(Refer Slide Time: 16:47)



```
#include <petsc.h>

int main(int argc, char**argv)
{


return PetscFinalize();
}
```

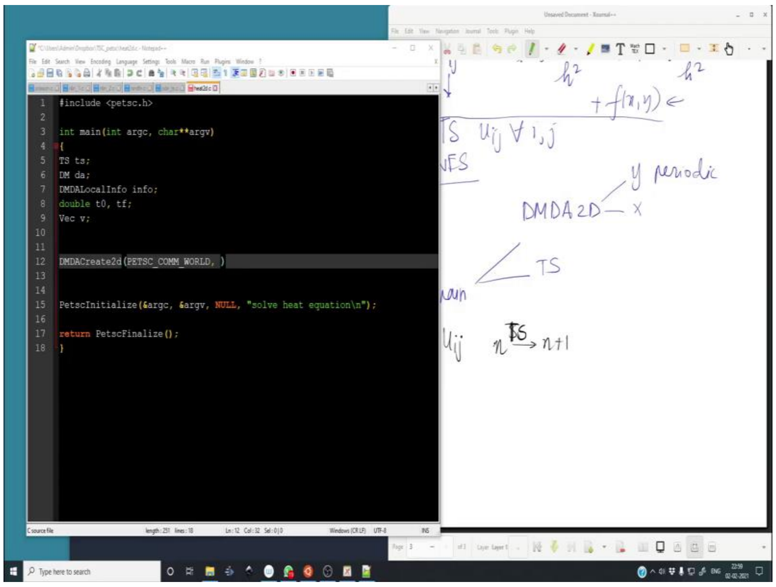
Let us first create the DMDA, then we will look into the time stepper. So, first things first we must do hash include petsc dot h. So, this is standard and after this we going to define main. So, int main, we have int argc char star star argv, then we must do petscFinalize, ok. This is just how we have defined it so far. So, we have to return this.

(Refer Slide Time: 17:31)



Let me save this file. So, let me call it heat2d dot c. So, finally, we have to do petsc finalize, we have to also do a petsc initialize somewhere.

(Refer Slide Time: 17:42)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, )


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

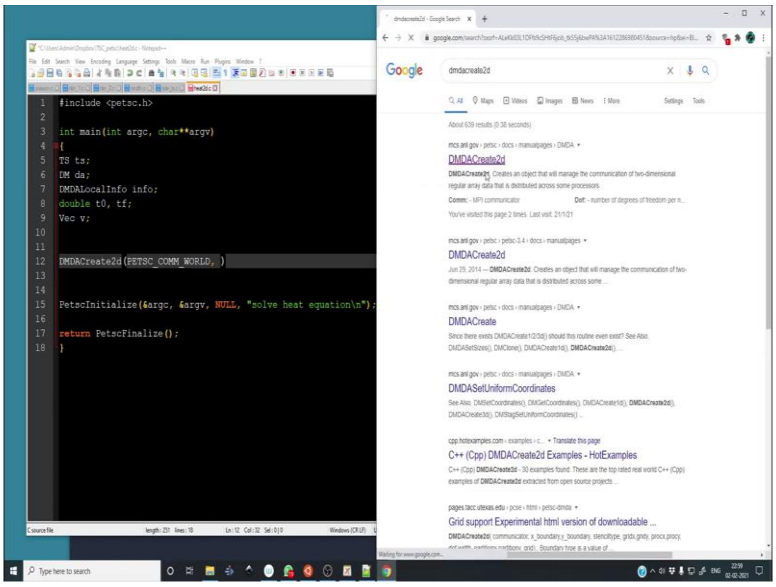
So, let us write it down petscInitialize. So, it takes argument some percent argc, some percent argv, then the null, then we must say some help string solve heat equation slash n, ok. So, we have the petsc initialization. So, we have to define certain variables. So, time step has to be defined TS ts, DM da has to be defined, so DM da, then DMDALocal

Info because we will need the local info to loop over various terms info then double we have to define the time span, alright.

What else do we need to do? We need to define the solution vector like we need to define the matrix; we will we do not need to define the matrix in main because we were passing everything to the time stepper and everything will be taken up by the time stepper. So, we do not need to define anything else and this should be sufficient to get started, ok.

So, let us set up the DMDA. So, DMDACreate2d, so this accepts the communicator. Apart from this, we will need to specify what the boundaries are. So, let me just open the function reference. So, DMDA create 2d it will; so, we have done this before, but still I just want to show you what the function reference is.

(Refer Slide Time: 19:40)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, )


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

dmdacreate2d

About 639 results (0.38 seconds)

mcs.anl.gov › petsc › docs › manualpages › DMDA
DMDACreate2d
DMDACreate2d. Creates an object that will manage the communication of two-dimensional regular array data that is distributed across some processors.
Comm: - MPI communicator    Dof: - number of degrees of freedom per n...
You've visited this page 2 times. Last visit: 21/1/21

mcs.anl.gov › petsc › petsc-3.4 › docs › manualpages
DMDACreate2d
Jun 29, 2014 — DMDACreate2d. Creates an object that will manage the communication of two-dimensional regular array data that is distributed across some ...

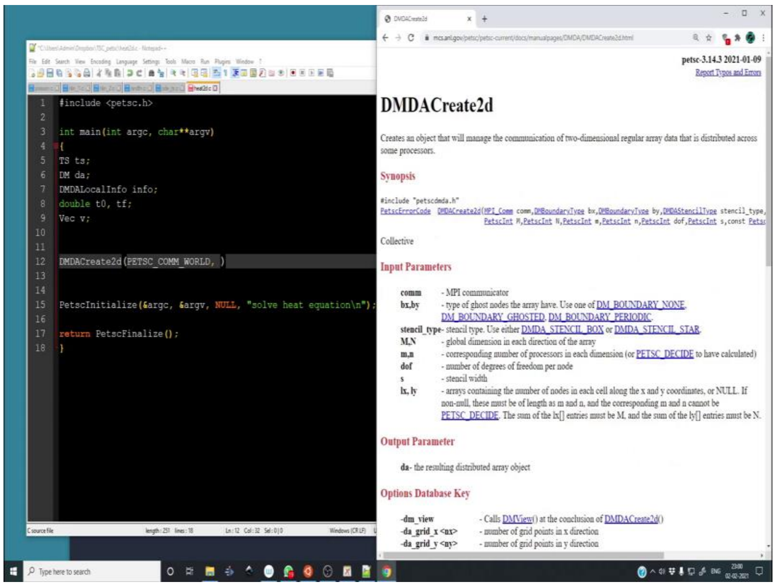
mcs.anl.gov › petsc › docs › manualpages › DMDA
DMDACreate
Since there exists DMDACreate1/2/3d() should this routine even exist? See Also. DMDASetSizes(), DMClone(), DMDACreate1d(), DMDACreate2d(), ...

mcs.anl.gov › petsc › docs › manualpages › DMDA
DMDASetUniformCoordinates
See Also. DMSetCoordinates(), DMGetCoordinates(), DMDACreate1d(), DMDACreate2d(), DMDACreate3d(), DMStagSetUniformCoordinates() ...

cpp.hotexamples.com › examples › c... Translate this page
C++ (Cpp) DMDACreate2d Examples - HotExamples
C++ (Cpp) DMDACreate2d - 30 examples found. These are the top rated real world C++ (Cpp) examples of DMDACreate2d extracted from open source projects ...

pages.tacc.utexas.edu › pcse › html › petsc-dmda
Grid support Experimental html version of downloadable ...
DMDACreate2d( communicator, x_boundary,y_boundary, stenciltype, gridx,gridy, procx,procy, dof,width, partitionx,partitiony, grid); Boundary type is a value of ...

(Refer Slide Time: 19:49)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, )


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

DMDACreate2d

Creates an object that will manage the communication of two-dimensional regular array data that is distributed across some processors.

Synopsis

```
#include "petscdmda.h"
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 Petsc
```

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 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. |

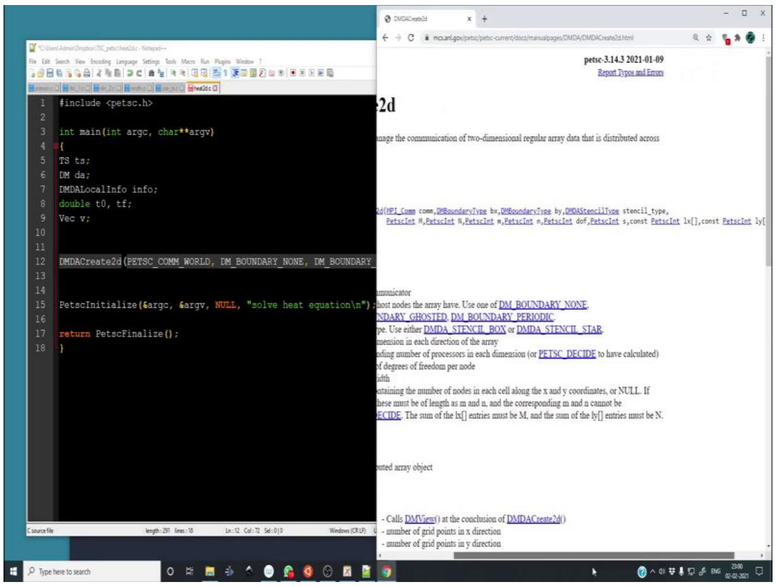
Output Parameter

da- the resulting distributed array object

Options Database Key

| | |
|---|---|
| -dm_view | - Calls DMView() at the conclusion of DMDACreate2d() |
| -da_grid_x <nx> | - number of grid points in x direction |
| -da_grid_y <ny> | - number of grid points in y direction |

(Refer Slide Time: 19:51)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

2d

nage the communication of two-dimensional regular array data that is distributed across

```
2d(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[
```

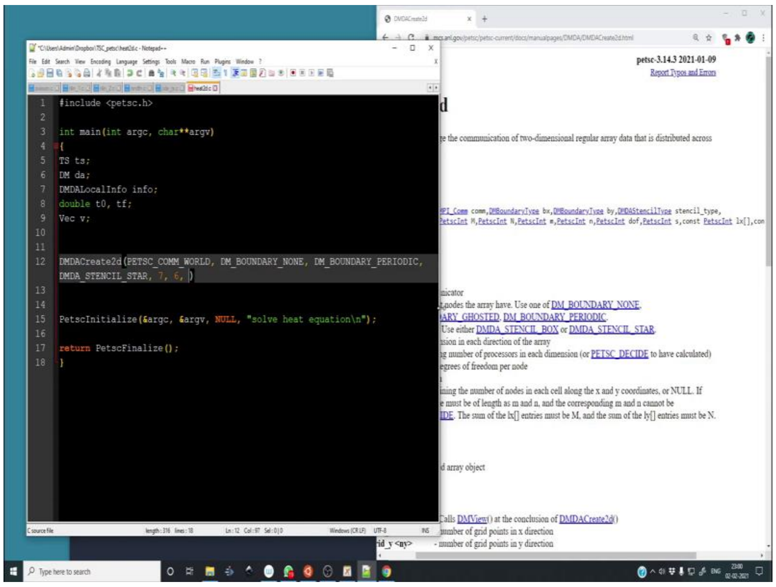
nmunicator
host nodes the array have. Use one of DM_BOUNDARY_NONE,
NDARY_GHOSTED, DM_BOUNDARY_PERIODIC.
pe. Use either DMDA_STENCIL_BOX or DMDA_STENCIL_STAR.
nension in each direction of the array
nding number of processors in each dimension (or PETSC_DECIDE to have calculated)
f degrees of freedom per node
idth
ntaining the number of nodes in each cell along the x and y coordinates, or NULL. If
hese must be of length as m and n, and the corresponding m and n cannot be
ECIDE. The sum of the lx[] entries must be M, and the sum of the ly[] entries must be N.

uted array object

- Calls DMView() at the conclusion of DMDACreate2d()
- number of grid points in x direction
- number of grid points in y direction

So, DMDAcreate2d, well there is a PDF version of all this, but I prefer to have the e, the online version it is more convenient. So, we have the communicator, then boundary type, so boundary type in X direction is going to be none.

(Refer Slide Time: 20:05)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, )


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

d

e the communication of two-dimensional regular array data that is distributed across

PI_Comm comm,DMBoundaryType bx,DMBoundaryType by,DMDAStencilType stencil_type,
PetscInt M,PetscInt N,PetscInt m,PetscInt n,PetscInt dof,PetscInt s,const PetscInt lx[],con

nicator
t nodes the array have. Use one of DM_BOUNDARY_NONE,
ARY_GHOSTED, DM_BOUNDARY_PERIODIC.
Use either DMDA_STENCIL_BOX or DMDA_STENCIL_STAR.
sion in each direction of the array
g number of processors in each dimension (or PETSC_DECIDE to have calculated)
egrees of freedom per node

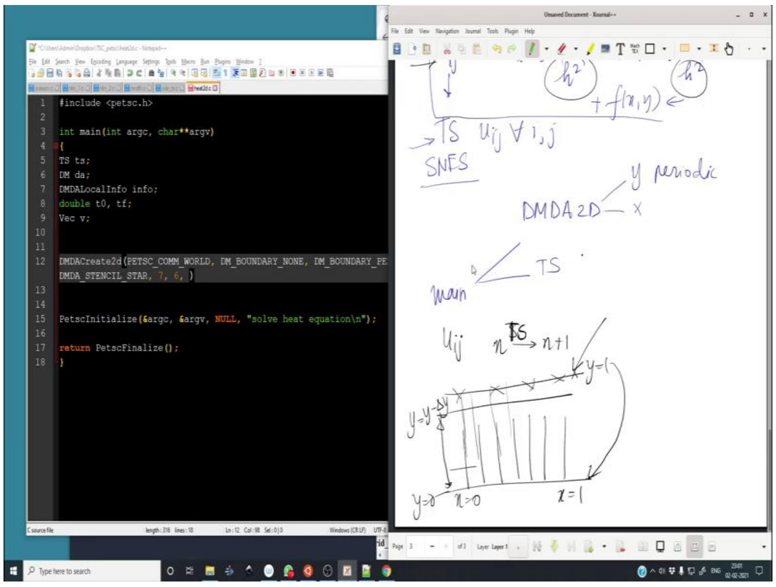
ining the number of nodes in each cell along the x and y coordinates, or NULL. If
e must be of length as m and n, and the corresponding m and n cannot be
IDE. The sum of the lx[] entries must be M, and the sum of the ly[] entries must be N.

d array object

Calls DMView() at the conclusion of DMDACreate2d()
umber of grid points in x direction
- number of grid points in y direction

So, DM BOUNDARY NONE, in the Y direction it is going to be PERIODIC, then we have to tell the kind of stencil we are going to have, so stencil type. So, we will have a star stencil, nothing, we do not need anything else. So, it is going to be DMDA STENCIL STAR that is the typical grid that we have. Then, we have to define the number of grid points. So, the number of grid points be 7, and in the other direction let it be 6.

(Refer Slide Time: 20:51)

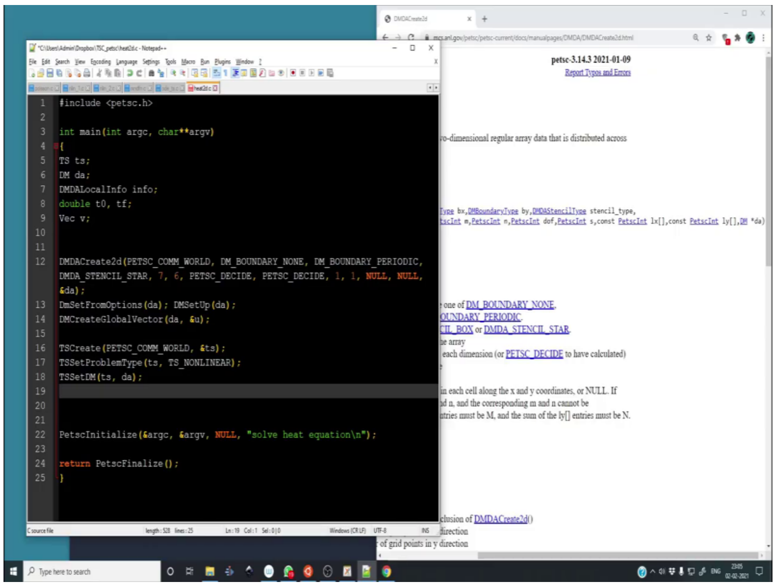


```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PE
DMDA_STENCIL_STAR, 7, 6, )


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

So, the reason why I have not chosen both to be 7 is that, once you make the boundary periodic you automatically do not go over all the grid points meaning 1, 2, 3, 4, 5, 6, 7. But if you; so, the x equal to 0 and x equal to 1 are included in this particular boundary none. But when you do periodic you are going from y equal to 0, to y equal to the second last step because y equal to 1 is tantamount to y equal to 0, so you do not have to save this.

So, you go, you have 1 less grid if you have an equal step, so because we are keeping the step equal we are keeping them both as h, right because they both are h, we will not have the final y line over here. We will simply go from 0 to y minus delta y or y minus h if you want. So, that is why you when you want 6, we need 1 less point if the boundaries are periodic. Then, we have to tell how many which processor, how many sort of grids it is going to handle.

(Refer Slide Time: 21:59)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL,
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);



PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

o-dimensional regular array data that is distributed across

```
Type bx,DMBoundaryType by,DMDAStencilType stencil_type,
tscInt m,PetscInt n,PetscInt dof,PetscInt s,const PetscInt lx[],const PetscInt ly[],DM *da)
```

one of DM_BOUNDARY_NONE,
OUNDARY_PERIODIC.
CIL_BOX or DMDA_STENCIL_STAR.
he array
each dimension (or PETSC_DECIDE to have calculated)

in each cell along the x and y coordinates, or NULL. If
d n, and the corresponding m and n cannot be
ntries must be M, and the sum of the ly[] entries must be N.

clusion of DMDACreate2d()
direction
of grid points in y direction

So, we can let petsc decide. So, it is m comma n. So, we can say let petsc decide that. Again for the Y direction; then after deciding all this we need to tell the number of degrees of freedom under stencil width, so it is going to be 1 comma 1. And finally, we need to tell the lx and ly, so they are typically kept null. So, if you do not do this, then you will have the number of nodes different than 1 in the x and y coordinates between two grids, ok.

So, you have NULL and then finally, you have to tell another null which is; so, this the second NULL is for. So, we have degrees of freedom as 1, then s was the stencil width which was 1, lx and ly both are null. Finally, the da, so we must pass the address of da. So, the address of da will be simply da.

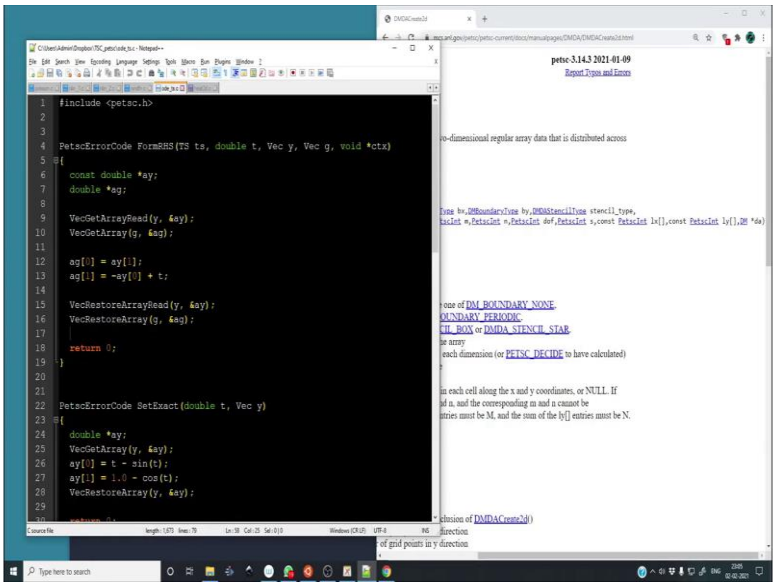
So, this creates the grid. So, once we have created the grid we can finish the formation of the grid and we have done this in the previous one of the previous lectures as well. So, DmsetFrom options because in case we pass some command line arguments, then we have DMSetUpda. Then, after setting up da we have DMCreateGlobalVector over da and that is the solution that we want, ok.

So, we have assigned all this. Now, we can move on to the time stepper specification, ok. So, let us proceed to the time stepping part. So, TSCreate again PETSC COMM WORLD and you have to pass the address of TS, then TSSetProblemType ts, TS NONLINEAR, ok. So, the problem is linear in u, but despite that we can keep the solver to the non-linear as we have done in the previous example.

After this then let us link the time stepper in the DMDA. So, it is TSSetDM, and we have to link ts and da. Then we have to do; so, what else is left? So, apart from, so the last example that we did was for ordinary differential equations. In this case, we need to sort of link the time stepper and the dm.

So, one additional thing that you need to do is set the RHS function, I mean based on the DMDA because in the previous example we were not linking into the DMDA, RHS function was defined as such, ok.

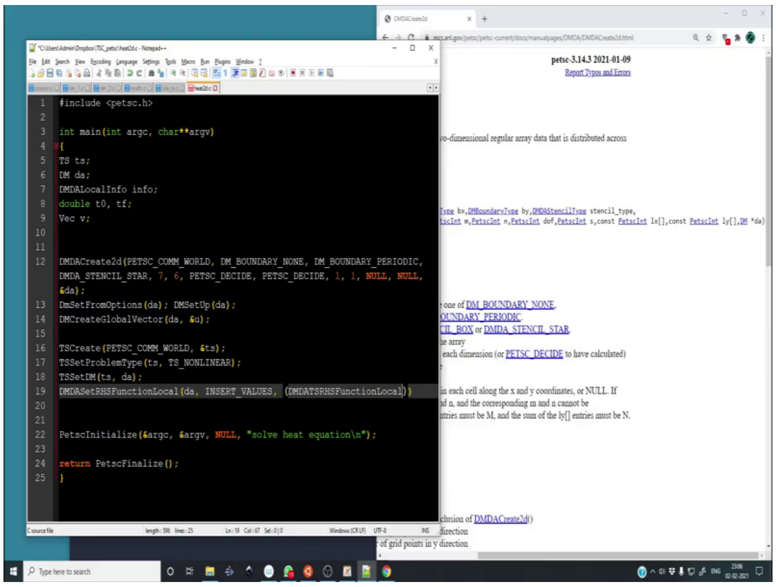
(Refer Slide Time: 25:49)



```c
#include <petsc.h>


PetscErrorCode FormRHS(TS ts, double t, Vec y, Vec g, void *ctx)
{
  const double *ay;
  double *ag;

  VecGetArrayRead(y, &ay);
  VecGetArray(g, &ag);

  ag[0] = ay[1];
  ag[1] = -ay[0] + t;

  VecRestoreArrayRead(y, &ay);
  VecRestoreArray(g, &ag);

  return 0;
}


PetscErrorCode SetExact(double t, Vec y)
{
  double *ay;
  VecGetArray(y, &ay);
  ay[0] = t - sin(t);
  ay[1] = 1.0 - cos(t);
  VecRestoreArray(y, &ay);

```

If you look into the previous example, the form RHS was just to set the RHS function, but in this case the RHS function is based on the DMDA.

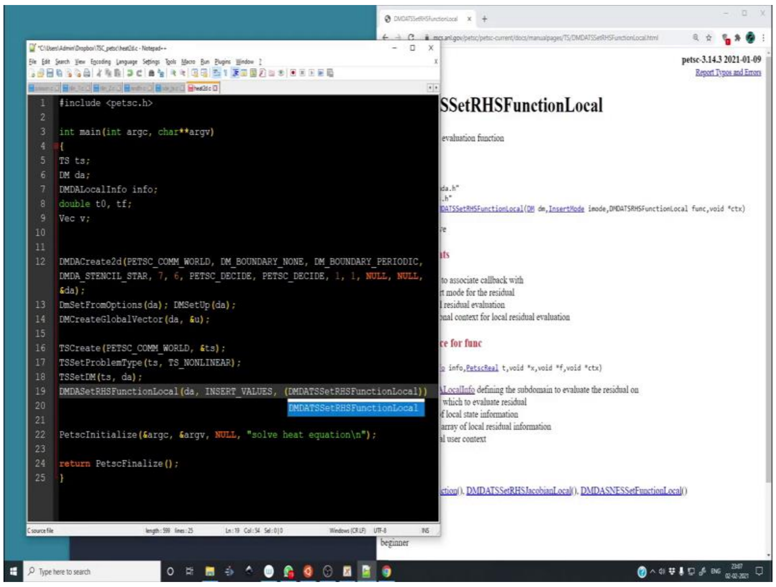
(Refer Slide Time: 26:03)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL,
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal))


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

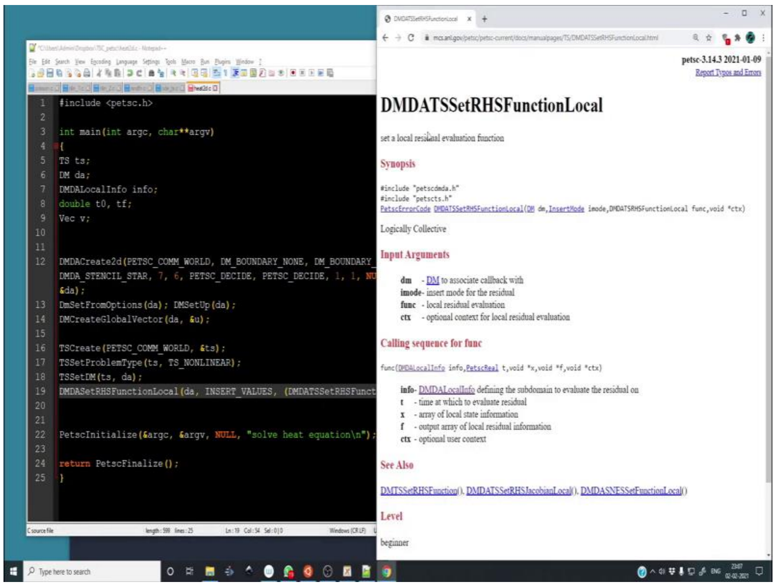
So, we must do TS not TS, DMDASetRHSFunctionLocal because it is based on a local grid, da, INSERT VALUES. And we have to now pass the function which will help us in creation of the RHS function, so the type of calling will have to be cast into the form DMDATSRHSFunction local.

(Refer Slide Time: 26:41)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL,
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunctionLocal))
                                           DMDATSSetRHSFunctionLocal

PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

(Refer Slide Time: 26:52)



```c
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NU
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunct


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

DMDATSSetRHSFunctionLocal

set a local residual evaluation function

Synopsis

```c
#include "petscdmda.h"
#include "petscts.h"
PetscErrorCode DMDATSSetRHSFunctionLocal(DM dm,InsertMode imode,DMDATSRHSFunctionLocal func,void *ctx)
```

Logically Collective

Input Arguments

**dm** - DM to associate callback with
**imode**- insert mode for the residual
**func** - local residual evaluation
**ctx** - optional context for local residual evaluation

Calling sequence for func

```c
func(DMDALocalInfo info,PetscReal t,void *x,void *f,void *ctx)
```

**info**- DMDALocalInfo defining the subdomain to evaluate the residual on
**t** - time at which to evaluate residual
**x** - array of local state information
**f** - output array of local residual information
**ctx** - optional user context

See Also

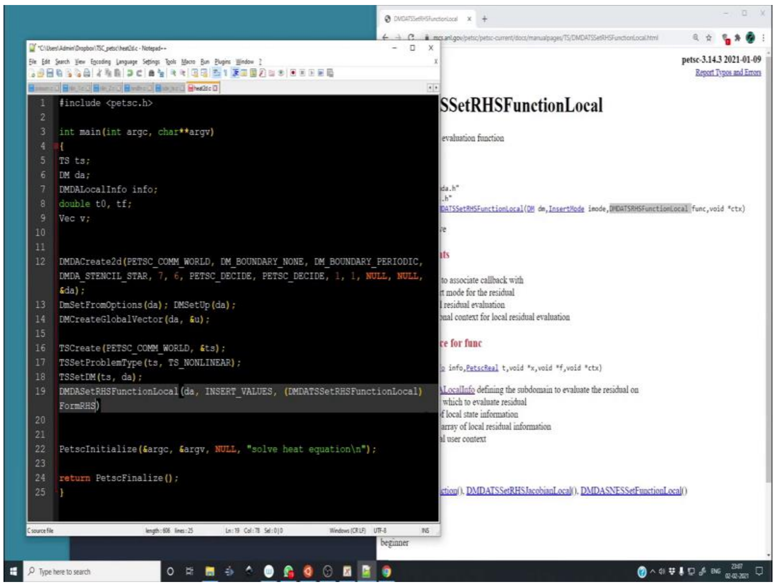
DMTSSetRHSFunction(), DMDATSSetRHSJacobianLocal(), DMDASNESSetFunctionLocal()

Level

beginner

So, if we go into the help of this on the reference file. So, DMDATSSetRHSFunction, ok. So, it is like a local residual evaluation function and it has to be done by passing this kind of a data type. So, we have to cast the function to this particular form that is why we are we are casting it.

(Refer Slide Time: 27:21)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL,
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunctionLocal)
FormRHS)


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

SSetRHSFunctionLocal

evaluation function

da.h"
.h"
DATSSetRHSFunctionLocal(DM dm,InsertMode imode,DMDATSRHSFunctionLocal func,void *ctx)

re

ts

to associate callback with
t mode for the residual
l residual evaluation
onal context for local residual evaluation

e for func

info,PetscReal t,void *x,void *f,void *ctx)

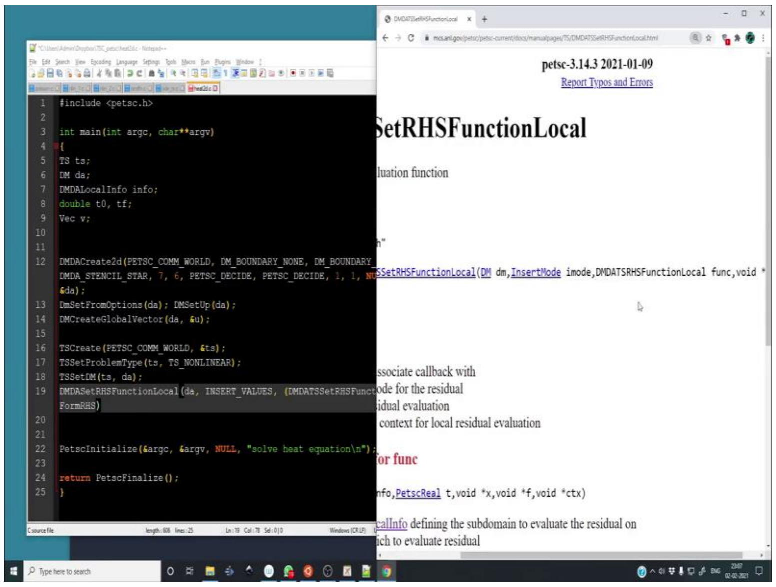
ALocalInfo defining the subdomain to evaluate the residual on
which to evaluate residual
f local state information
array of local residual information
al user context

ction(), DMDATSSetRHSJacobianLocal(), DMDASNESSetFunctionLocal()

beginner

So, this is to cast whatever function we have. So, let the function be form RHS, no problem. So, we have to create a function form RHS which will help us in creating the RHS function and it has to be typecast to the form DMDARHSFunctionalLocal, RHS function local.

(Refer Slide Time: 27:35)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, N
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunct
FormRHS)


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n")

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

SetRHSFunctionLocal

luation function

h"
SSetRHSFunctionLocal(DM dm,InsertMode imode,DMDATSRHSFunctionLocal func,void *

ssociate callback with
ode for the residual
idual evaluation
context for local residual evaluation

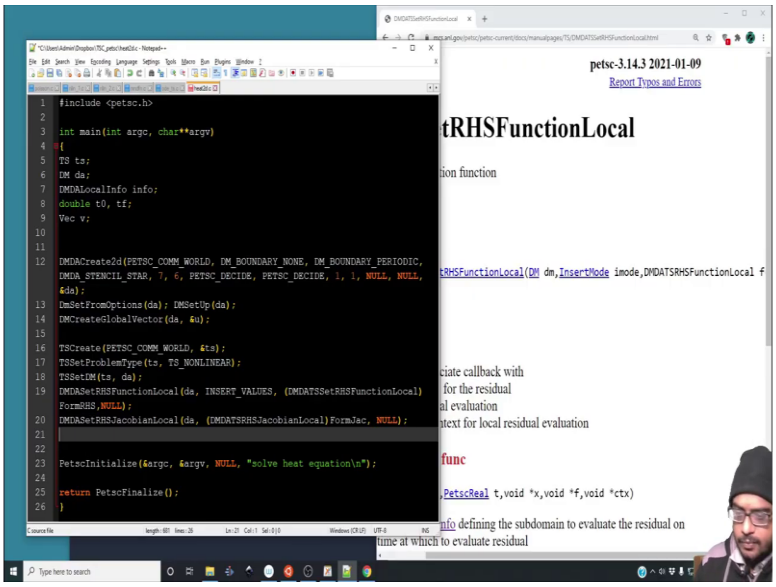
or func

nfo,PetscReal t,void *x,void *f,void *ctx)

calInfo defining the subdomain to evaluate the residual on
ich to evaluate residual

So, this is called as a type casting to this form, this is where we are typecasting it to that form. So, apart from this we need to provide a function, the application context.

(Refer Slide Time: 27:47)



```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL,
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

tRHSFunctionLocal

tion function

```
tRHSFunctionLocal(DM dm,InsertMode imode,DMDATSRHSFunctionLocal f
```

ciate callback with
for the residual
al evaluation
ntext for local residual evaluation

func

```
,PetscReal t,void *x,void *f,void *ctx)
```

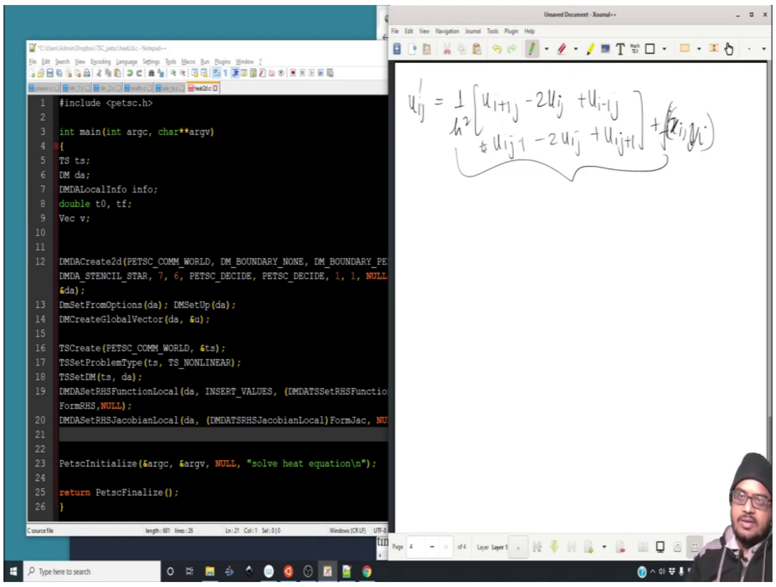
nfo defining the subdomain to evaluate the residual on
time at which to evaluate residual

In this case there is no application context, so this can be simply void star ctx. We are not going to use it anyway. So, this sets the RHS local function. Apart from this we must set the DMDA and the TS with the local Jacobian. So, we will do DMDASetRHSJacobianLocal and the calling sequence is again quite same, so it is da. Sort of inserting values we will simply have a function call back.

But instead of DMDATSSetRHSFunctionLocal. We will have DMDATSRHS JacobianLocal. So, we will change this. So, DMDA this will be RHS, instead of RHS function local it will be RHSJacobianLocal. And let me change this to FormJacobian, and well this cannot be void star ctx, it has to be just some ctx, it does not matter. We can pass a null over here, it does not matter. FormJac and then finally, NULL.

If we were to pass some variable from we were to have suppose if we define diffusivity in this problem and pass it to these functions, we would have to have an address to the that particular structure. And we have done something like this in the previous lectures. But we are not going to use it over here. We do not need to do that, ok. So, yeah. So, that defines the linking of the TS and the DMDA. So, we must go about defining what this RHS function in the Jacobian will be, alright.

(Refer Slide Time: 29:57)

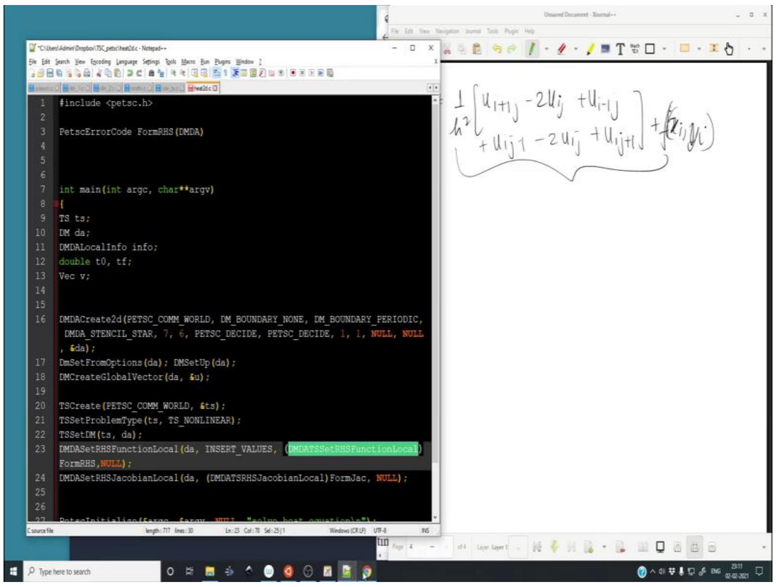


```
#include <petsc.h>

int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PE
DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL
&da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunctio
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NU


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

$$u_{ij}' = \underbrace{\frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_i)}$$

So, in this particular matter of lines, what do we have? So, we have u ij prime is equal to h not x square, 1 over x square times u i plus 1 j minus 2 u ij plus u i minus 1 j plus u ij minus 1 minus 2 u ij plus u ij plus 1 plus f of xi, yj. So, the RHS will be this entire thing and the Jacobian will be the derivative of this entire thing. It is as simple as that, ok.

(Refer Slide Time: 30:39)

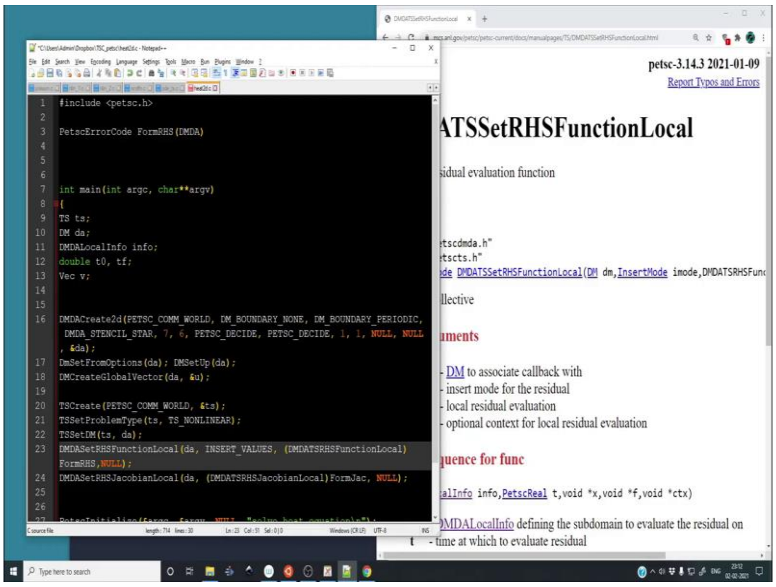


```
#include <petsc.h>

PetscErrorCode FormRHS(DMDA)



int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSSetRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);
```

$$\underbrace{\frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_i)}$$

So, let us first form the RHS local function. So, we define it over here. So, let us see error code. The name of the function is form RHS and the input to forming the RHS will be the DMDA. So, it will be DMDA wait; so, yeah. So, the calling sequence here, so we have to look into the calling sequence of this particular function type, so we must look into that. What is the calling sequence? So, we have to pass the DM, then we have to pass the mode of insertion, then the RHS function local. So, now let us see what RHS function, set RHS function, DMDA TS RHS Function Local; DMDA oops.

(Refer Slide Time: 32:09)



```
#include <petsc.h>

PetscErrorCode FormRHS(DMDA)



int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);
```

petsc-3.14.3 2021-01-09
Report Typos and Errors

ATSSetRHSFunctionLocal

sidual evaluation function

```
tscdmda.h"
tscts.h"
de DMDATSSetRHSFunctionLocal(DM dm,InsertMode imode,DMDATSRHSFun
```

llective

uments

- DM to associate callback with
- insert mode for the residual
- local residual evaluation
- optional context for local residual evaluation

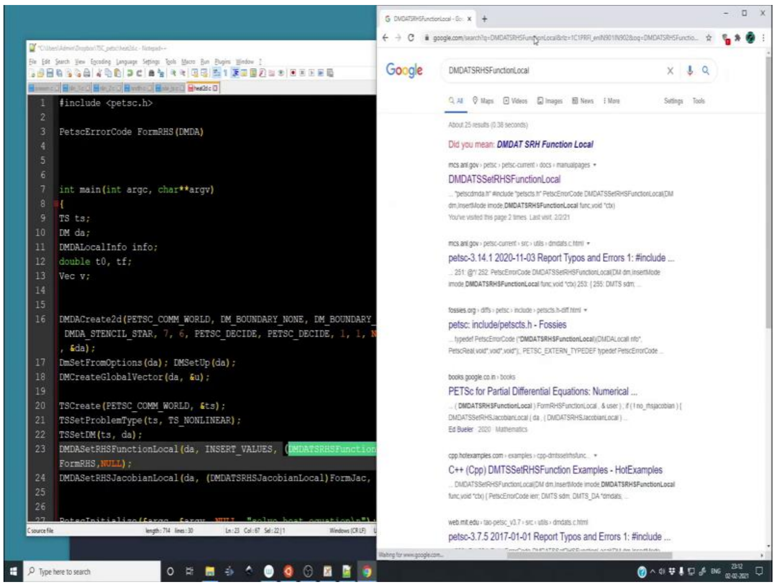
quence for func

```
alInfo info,PetscReal t,void *x,void *f,void *ctx)
```

DMDALocalInfo defining the subdomain to evaluate the residual on
t - time at which to evaluate residual

So, this was set RHS function, but this is not supposed to be that. This is supposed to be RHS function local. So, DMDATSRHSFunctionLocal, not set and yeah. So, let us now look into the call back sequence of this, ok.

(Refer Slide Time: 32:23)



```
#include <petsc.h>

PetscErrorCode FormRHS(DMDA)



int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, N
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunction
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac,
```

DMDATSRHSFunctionLocal

About 25 results (0.38 seconds)

Did you mean: *DMDAT SRH Function Local*

mcs.anl.gov › petsc › petsc-current › docs › manualpages
DMDATSSetRHSFunctionLocal
... "petscdmda.h" #include "petscts.h" PetscErrorCode DMDATSSetRHSFunctionLocal(DM dm,InsertMode imode,**DMDATSRHSFunctionLocal** func,void *ctx)
You've visited this page 2 times. Last visit: 2/2/21

mcs.anl.gov › petsc-current › src › utils › dmdats.c.html
petsc-3.14.1 2020-11-03 Report Typos and Errors 1: #include ...
... 251: @*/ 252: PetscErrorCode DMDATSSetRHSFunctionLocal(DM dm,InsertMode imode,**DMDATSRHSFunctionLocal** func,void *ctx) 253: { 255: DMTS sdm; ...

fossies.org › diffs › petsc › include › petscts.h-diff.html
petsc: include/petscts.h - Fossies
... typedef PetscErrorCode (***DMDATSRHSFunctionLocal**)(DMDALocalInfo*, PetscReal,void*,void*,void*); PETSC_EXTERN_TYPEDEF typedef PetscErrorCode ...

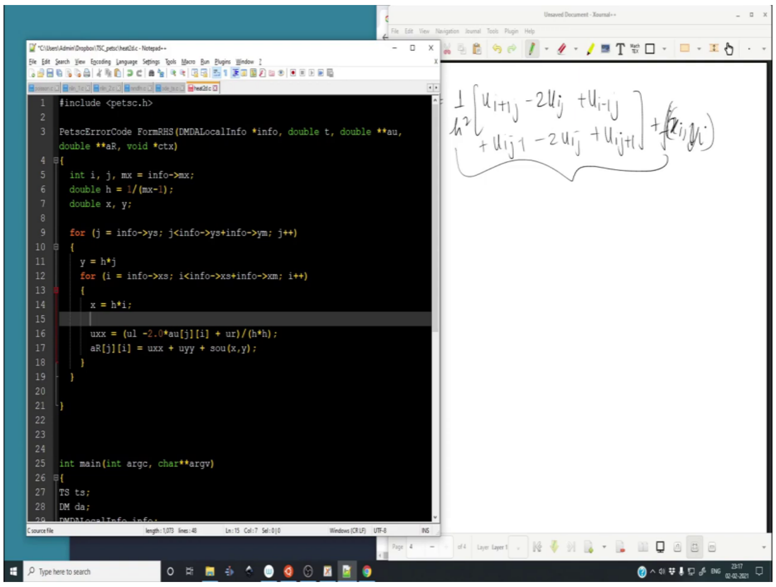
books.google.co.in › books
PETSc for Partial Differential Equations: Numerical ...
... ( **DMDATSRHSFunctionLocal** ) FormRHSFunctionLocal , & user ) ; if ( ! no_rhsjacobian ) { DMDATSSetRHSJacobianLocal ( da , ( DMDATSRHSJacobianLocal ) ...
Ed Bueler · 2020 · Mathematics

cpp.hotexamples.com › examples › cpp-dmtssetrhsfunc...
C++ (Cpp) DMTSSetRHSFunction Examples - HotExamples
... DMDATSSetRHSFunctionLocal(DM dm,InsertMode imode,**DMDATSRHSFunctionLocal** func,void *ctx) { PetscErrorCode ierr; DMTS sdm; DMTS_DA *dmdats; ...

web.mit.edu › tao-petsc_v3.7 › src › utils › dmdats.c.html
petsc-3.7.5 2017-01-01 Report Typos and Errors 1: #include ...

So, it is not given explicitly what that calling sequence is, but yeah, let me tell you how it is.

(Refer Slide Time: 32:35)



```c
#include <petsc.h>

PetscErrorCode FormRHS(DMDALocalInfo *info, double t, double **au,
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1/(mx-1);
  double x, y;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;

      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
      aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}



int main(int argc, char**argv)
{
TS ts;
DM da;
```

$$\underbrace{\frac{1}{h^2}\left[\begin{matrix} u_{i+1j} - 2u_{ij} + u_{i-1j} \\ + u_{ij1} - 2u_{ij} + u_{ij+1} \end{matrix}\right] + f}(x_i, y_i)$$

So, you have to pass the DMDALocalInfo as with the address of local info. So, info then you need double t for time and you need a double au. So, something similar we have done in the previous lecture. This is an auxiliary array to hold u so, that we can convert between the vector object in petsc with the auxiliary array object of c.

Then, we have double star star auxiliary array for the RHS function. So, let me call it aR and finally, the context of void star ctx, we do not need anything from this. So, let me define this function. So, first of all we need to loop counter. So, int i comma j, we are going to define h as; so, whatever the info is being passed we must define the step from it or let us say mx, info mx. So, this is what the x limit will be.

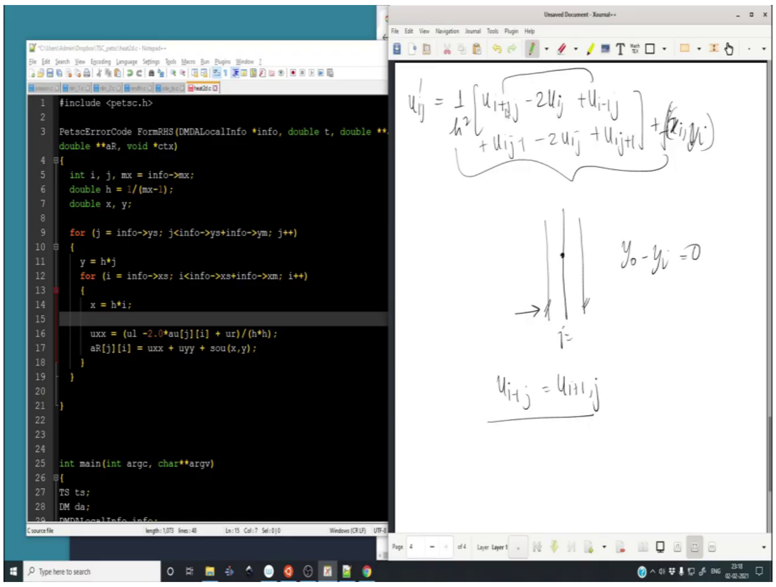
Then, what can we do? So, let us first get the spacings. So, the spacing in this case will be simply 1 over mx minus 1, ok. So, double h equal to 1 over mx minus 1 that is this grid spacing, ok so far so good. So, let us define double x comma y, yeah. So, let us now do a double loop for j equal to info ys, j less than info ys, plus info ym j plus plus. Let me copy this.

So, for, so instead of this we must have i. So, we will doing the inner loop over x, ok. We must be very careful when doing this kind of copy paste stuff, ok. So, after this we must define what y is. So, y is going to be simply h times j and x is going to be h times x, h times i not x. So, now let us apply the not let us apply the let us find out the RHS. So, the RHS will be ag not g, R; j comma i. So, in general it will be equal to that entire RHS term.

What is the random origin term? It is this term. So, it is going to be let us define it as uxx and uyy. So, uxx plus uyy plus source as a function of x and y. Now, what is uxx? So, uxx is going to be let us define in terms of ul as the u left, u left minus twice of and I will tell you why I am writing it in terms of ux minus 2.0 times au j comma i not j comma i, j i plus ur upon h square.

And the reason why I am defining in terms of ul and ur is because depending on whether you are sitting at the left most boundary or depending on whether you are sitting at the rightmost boundary. The ul and ur have to be appropriately changed with respect to the ghost nodes, that is because the right side boundary is a Neumann boundary condition.

(Refer Slide Time: 37:29)



```
#include <petsc.h>

PetscErrorCode FormRHS(DMDALocalInfo *info, double t, double **
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1/(mx-1);
  double x, y;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;

      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
      aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}



int main(int argc, char**argv)
{
TS ts;
DM da;
```

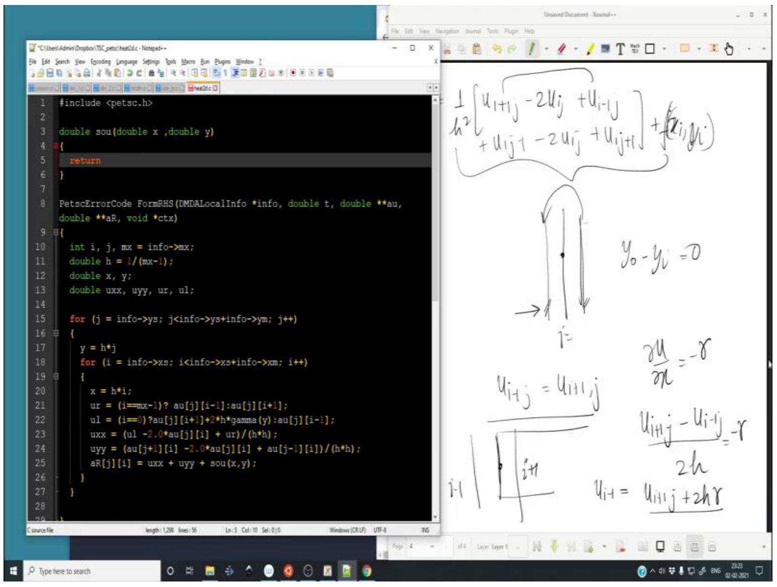
$$u'_{ij} = \frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_i)$$

$$y_0 - y_i = 0$$

$$u_{i-1j} = u_{i+1,j}$$

So, when i is equal to mx minus like rather, when your i is at the last boundary this minus 1 boundary and plus boundary they are related because it is a Neumann boundary condition. So, y outside minus y inside equal to 0, y outside equal to y inside. So, we must set for the innermost boundary u i minus 1 j is equal to u i plus 1 j and that is the case then these two become equal, ok; these two become equal. And so, the u left will be equal to u, right essentially, ok.

(Refer Slide Time: 38:20)



```c
#include <petsc.h>

double sou(double x ,double y)
{
  return
}

PetscErrorCode FormRHS(DMDALocalInfo *info, double t, double **au,
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i-1]:au[j][i+1];
      ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
      uyy = (au[j+1][i] -2.0*au[j][i] + au[j-1][i])/(h*h);
      aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}
```

$$\frac{1}{h^2}\begin{bmatrix} u_{i+1j} - 2u_{ij} + u_{i-1j} \\ + u_{ij1} - 2u_{ij} + u_{ij+1} \end{bmatrix} + f(x_i, y_i)$$

$y_0 - y_i = 0$

$i =$

$u_{i-1j} = u_{i+1,j}$

$\frac{\partial u}{\partial x} = -\gamma$

$\frac{u_{i+1j} - u_{i-1j}}{2h} = -\gamma$

$i-1$ $i+1$

$u_{i-1} = u_{i+1j} + 2h\gamma$

So, the way to do it is ul is equal to, so is i equal to equal to mx minus 1. So, this is a compact way of writing an fl statement. So, if it is equal then you say au j comma i minus 1. So, ul will become if it is true, then it if it is true then the ghost node will be equal to the innermost node. So, basically ur will be equal to ul.

So, if it is true, then you simply set ul by u ji plus 1 because the left boundary will sort of get the value of the sorry; we are we are looking at the rightmost boundary, so this will be u i minus 1, that is the rightmost boundary, this has to be ur, ok. So, when you are sitting at the rightmost boundary then ur; if it is this if this statement is true then you are sitting at the rightmost boundary.

If that is true then ur has to be given the left value which is this otherwise you simply assign it with au j comma i plus 1. So, this is for then it is not at the outermost boundary then ur will be simply u ji plus 1. If it is sitting at the outermost boundary then it has to be equal to ul which in this case is going to be u ji plus 1.

Similarly, if i is equal to 0, meaning you are sitting on the left most boundary, then the left outer point will be equal to the right point. So, if it is true then it has to be equal to u i, au j i plus 1 that is the left value will actually be the innermost point that is to say this is at the left boundary.

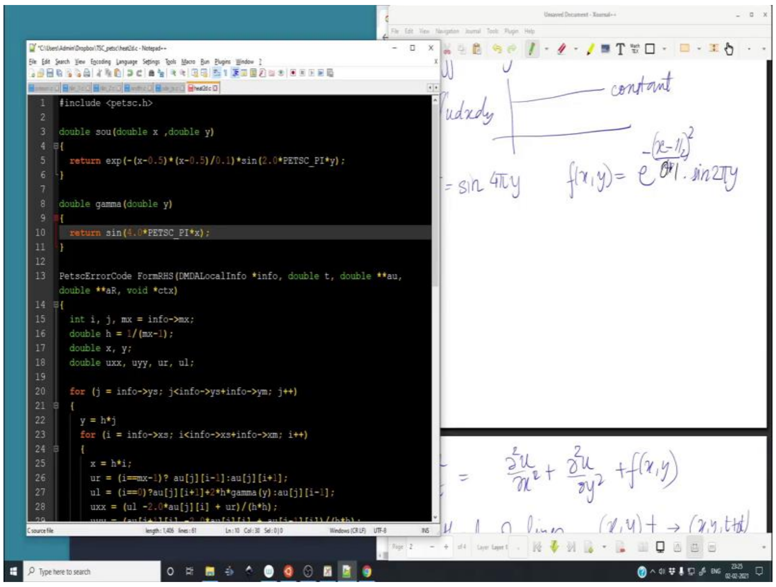
This is i plus 1, this is i minus 1 and because del u del x is equal to minus gamma, so u i plus 1 j minus u i minus 1 j upon 2h is equal to minus gamma. So, if this is true then u i minus 1 will be equal to u i plus 1 j plus 2h gamma, ok. So, we have to assign that simply. So, it will be u i plus 1 plus twice of h times the gamma, and in this case let us just write as gamma of y; if it is not then it is simply au j comma i minus 1, ok.

So, this is how you define ul and ur based on this you are defining uxx, based on uxx and uyy; so, uyy is not complicated at all because there is no special cases for it because the top and the bottom boundaries are periodic. We can simply copy this, paste this, and instead of ul and ur we simply can put au j plus 1 i plus au j minus 1 i. So, it is as simple as that, ok. So, now we must declare these variables which we have used. So, we have made use of uxx, uyy, ur, ul.

So, apart from this we must define two additional functions one is for the Neumann boundary condition gamma and one is for the source term. So, let us define it, alright. So, here we will write the source and the gamma, ok.

And they they are easy functions, but nevertheless we must define them. So, we will say double source x comma; so, it will be double x, double y. So, this function simply has to return the source term which we had defined already as this e to the power minus of whatever it is.

(Refer Slide Time: 43:30)



```
#include <petsc.h>

double sou(double x ,double y)
{
  return exp(-(x-0.5)*(x-0.5)/0.1)*sin(2.0*PETSC_PI*y);
}

double gamma(double y)
{
  return sin(4.0*PETSC_PI*x);
}

PetscErrorCode FormRHS(DMDALocalInfo *info, double t, double **au,
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i-1]:au[j][i+1];
      ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
```

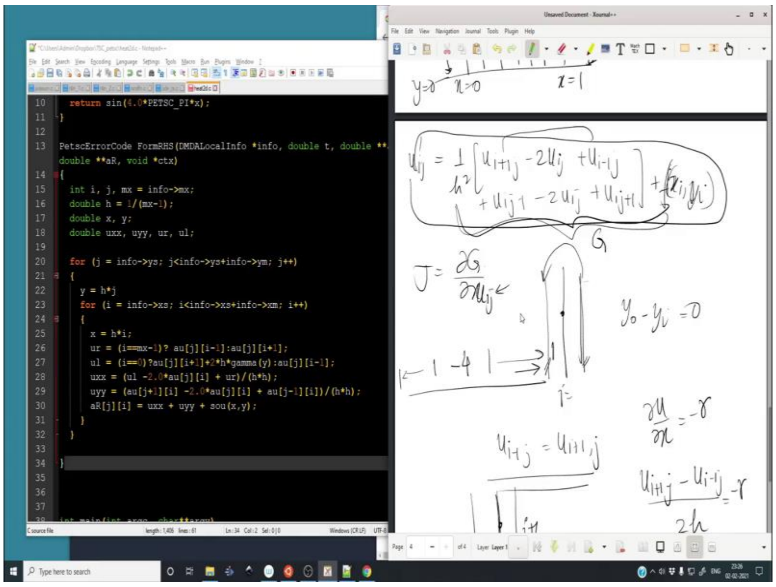
constant
udxdy
= sin 4πy
$f(x,y) = e^{-\frac{(x-1/2)^2}{0.1}} \cdot \sin 2\pi y$
$= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x,y)$
$(x,y)t \rightarrow (x,y,t+\delta t)$

So, we can write exponential of minus of x minus 0.5 times, x minus 0.5. So, this x square divided by; let us not take it 4 because that will be a very thick Gaussian. Let us make it slightly thinner. Let us say it is 0.1. It does not matter anyway because the conditions are already being satisfied for the solubility. So, divided by 0.1 times sin of 2 into 2.0 into PETSC pi that is a macro for pi times y.

So, simply have to return this. Similarly, we have to define the gamma as well. So, define, so the return value will be double it will be gamma of double y. We must return sin 4 by x, so sin 4.0 times PETSC pi times x. So, there you have; we have the two functions also defined.

So far we have defined the right hand side, now based on this let us form the; so, we have form the local function, you have to form the Jacobian as well. Jacobian is also quite simple to construct. So, in this discretized form the Jacobian, so what do we have? So, this is what the function of this is what the function is. So, this is the G function that we have used in the previous example.

(Refer Slide Time: 45:26)

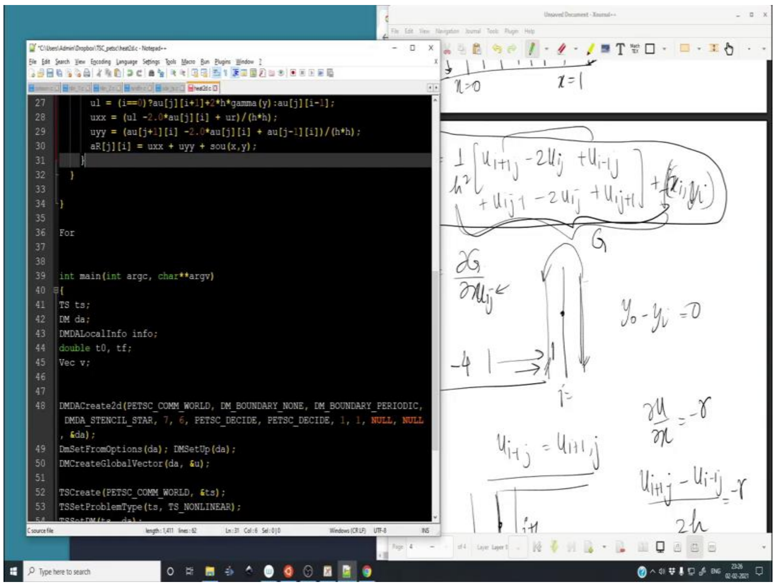


```
  return sin(4.0*PETSC_PI*x);
}

PetscErrorCode FormRHS(DMDALocalInfo *info, double t, double **
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i-1]:au[j][i+1];
      ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
      uyy = (au[j+1][i] -2.0*au[j][i] + au[j-1][i])/(h*h);
      aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}
```

$y=0$ $x=0$ $x=1$
$u_{ij}' = \frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_i)$
$G$
$J = \frac{\partial G}{\partial u_{ij}}$
$y_0 - y_i = 0$
$1 \quad 1 \quad -4 \quad 1 \quad 1$
$i=$
$\frac{\partial u}{\partial x} = -\gamma$
$u_{i-1j} = u_{i+1,j}$
$\frac{u_{i+1j} - u_{i-1j}}{2h} = -\gamma$
$i+1$

So, the Jacobian will be del G i del x j and this x is not to become confused with all of that, it is basically del G del x del u ij. So, here the Jacobian will be minus 4 because when you take a derivative with respect to the variable u ij, this is 1d minus 4, then you have 1, 1, then an offset with 1 and then offset with 1, alright. So, the Jacobian has that pentadiagonal structure and it is also not that complicated to set up. So, let us set it up, let us set up the how the Jacobian will be evaluated.

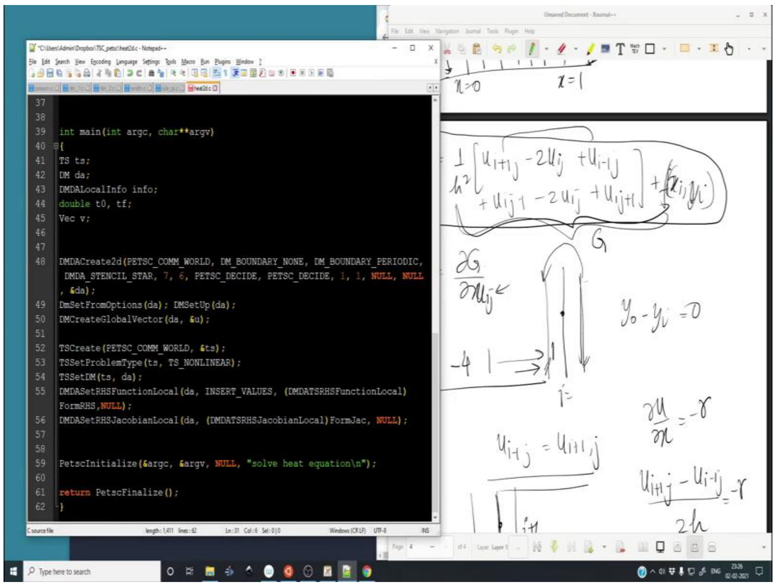
(Refer Slide Time: 46:07)



```
        ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
        uxx = (ul -2.0*au[j][i] + ur)/(h*h);
        uyy = (au[j+1][i] -2.0*au[j][i] + au[j-1][i])/(h*h);
        aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}

For


int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
```

$x=0$
$x=1$
$\frac{1}{h^2}\left[u_{i+1j}-2u_{ij}+u_{i-1j}+u_{ij-1}-2u_{ij}+u_{ij+1}\right]+f(x_i,y_i)$
$G$
$\frac{\partial G}{\partial u_{ij}}$
$-4$
$y_0-y_i=0$
$\frac{\partial u}{\partial x}=-\gamma$
$u_{i+1j}=u_{i+1,j}$
$\frac{u_{i+1j}-u_{i-1j}}{2h}=-\gamma$

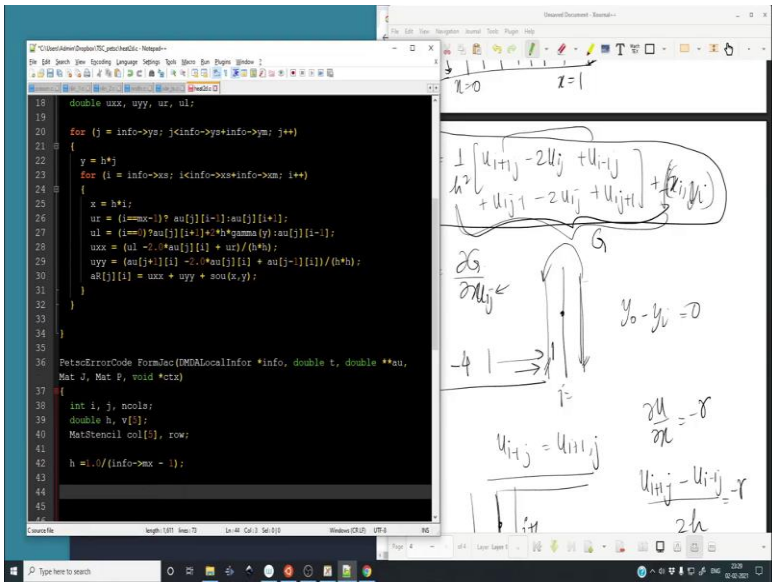
(Refer Slide Time: 46:11)



```


int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);


PetscInitialize(&argc, &argv, NULL, "solve heat equation\n");

return PetscFinalize();
}
```

So, let us define it form, so the name of the function was FormJac.

(Refer Slide Time: 46:16)



```c
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->ys+info->ym; j++)
  {
    y = h*j
    for (i = info->xs; i<info->xs+info->xm; i++)
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i-1]:au[j][i+1];
      ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
      uxx = (ul -2.0*au[j][i] + ur)/(h*h);
      uyy = (au[j+1][i] -2.0*au[j][i] + au[j-1][i])/(h*h);
      aR[j][i] = uxx + uyy + sou(x,y);
    }
  }

}

PetscErrorCode FormJac(DMDALocalInfor *info, double t, double **au,
Mat J, Mat P, void *ctx)
{
  int i, j, ncols;
  double h, v[5];
  MatStencil col[5], row;

  h =1.0/(info->mx - 1);

```

So, the return value will be a PetscErrorCode Formjac, J with the capital. So, FromJac, so the inputs to this function will be quite similar to what we have for the function call back, but we have just defined in the previous function. So, it has to be a DMDALocalInfor, star info double t double star star au, Mat J, Mat P, in case you have a different precondition for the Jacobian and then void star ctx, ok.

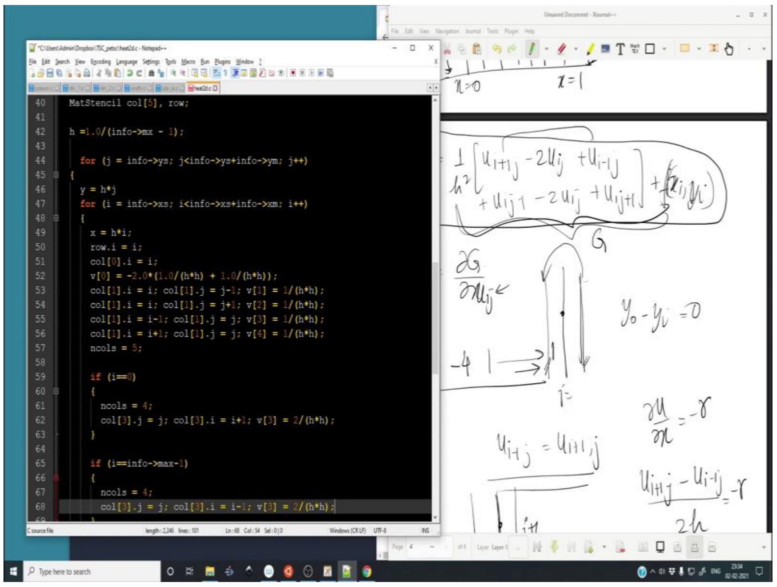
So, let us define it. So, it is going to be having the similar structure as for the previous program. So, we must have two counters. So, int i comma y, i comma j. And we must also have an integer as the number of columns that we want to insert, so it will be ncols. Apart from this we must have double h because we need the grid spacing, then we need the number of values to be inserted.

So, at max you will have 5 values inserted because over here we have one main diagonal, then offset, another offset, another offset by a large, number another offset on the right by large number. So, we have essentially 5 values which we can input. So, we define and define an array with size 5 these are the values array.

Then, we need a data type of the type MatStencil because it is quite specific to the construction of the array. So, oops. So, MatStencil has the same data type as int. It does not matter if you use this, but it is a legitimate data type in petsc. You can choose to use it or not to use it you can simply use int as well, alright. So, h is equal to 1 over, so 1.0

over, so this has to be actually 1.0, ok. So, 1.0 divided by info mx minus 1. So, that is the grid spacing. Now, we are in a position to perform the double loop.

(Refer Slide Time: 49:09)



```c
MatStencil col[5], row;

h =1.0/(info->mx - 1);

  for (j = info->ys; j<info->ys+info->ym; j++)
{
  y = h*j
  for (i = info->xs; i<info->xs+info->xm; i++)
  {
    x = h*i;
    row.i = i;
    col[0].i = i;
    v[0] = -2.0*(1.0/(h*h) + 1.0/(h*h));
    col[1].i = i; col[1].j = j-1; v[1] = 1/(h*h);
    col[1].i = i; col[1].j = j+1; v[2] = 1/(h*h);
    col[1].i = i-1; col[1].j = j; v[3] = 1/(h*h);
    col[1].i = i+1; col[1].j = j; v[4] = 1/(h*h);
    ncols = 5;

    if (i==0)
    {
      ncols = 4;
      col[3].j = j; col[3].i = i+1; v[3] = 2/(h*h);
    }

    if (i==info->max-1)
    {
      ncols = 4;
      col[3].j = j; col[3].i = i-1; v[3] = 2/(h*h);
```

$x=0$
$x=1$
$\frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_j)$
G
$\frac{\partial G}{\partial u_{ij}}$
$y_0 - y_i = 0$
$-4$
$u_{i-1j} = u_{i+1,j}$
$\frac{\partial u}{\partial x} = -r$
$\frac{u_{i+1j} - u_{i-1j}}{2h} = -r$

So, we have, so let me copy the loop at the risk of making a silly mistake. Let me close the loop immediately, alright. So, we have defined what the double loop is going to be. Inside that double loop we have row dot i equal to i because we going to insert in the row number i. If you are looking at the variable u i comma j, then we have the different columns. So, call 0 dot j or rather call 0 dot i will also equal to i. So, row column is fixed. Now, we must fix the column variable.

So, column dot i and column dot j, ok. So, first of all what is the values that we want to input? It is going to be minus 2.0 times 1 over h star h plus 1.0 over h times h. This is also going to be 1.0. And this you can easily find out its coming from the differentiation of the Jacobian. So, one is coming from this term and one is coming from this, nothing else, ok.

So, the central value that we are going to insert. Now, coming to the columns, different columns; so, col 0 and not col 0, col 1 dot i is going to be i, col 1 dot j is going to be j minus 1. Similarly, let us copy this, oops, the 4 values. So, we have i j minus 1, then i j plus 1, then i minus 1, then i plus 1 and these are simply going to be j.

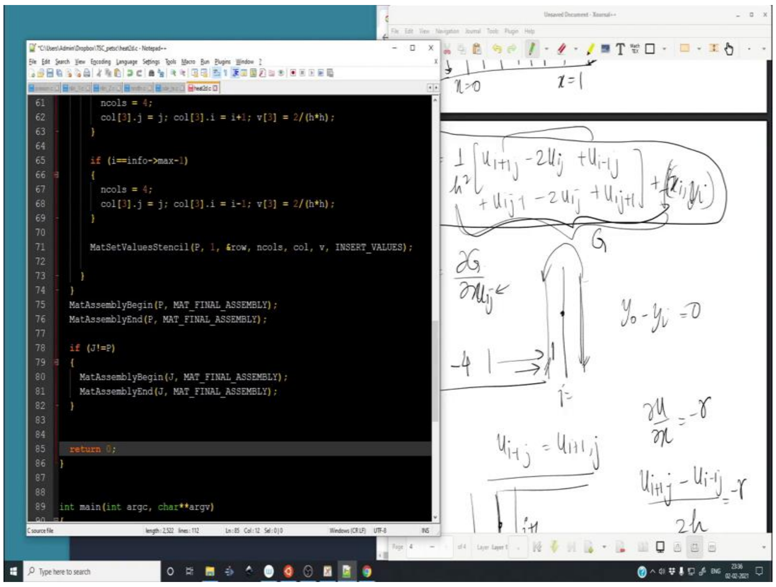
So, these are the locations where we want to insert the values for the Jacobian. And what will be the values? So, v 1 will be equal to 1 over h square. In fact, all the values which will be 1 over x square. So, let me copy this. So, this will be 2, 3, 4. So, unless we are at the boundary point, if you are at the boundary point, 2, 3, 4, correct. So, unless you are at the boundary point nothing like this will happen. So, we have n calls 5 because we want to insert all of this.

And if we are at the boundary, then what happens? If i equal to 0 and we are at the left boundary, then we will not have to insert 5 values instead we will have to insert only 4 values. And I want you to actually go ahead and find this out. So, we have to modify the column that is going to be inserted because this if the left boundary this is not going to be i minus 1 it is going to be something else.

So, it is going to be call 3 dot j will be equal to j and because of the Neumann boundary condition call 3 dot i will be equal to now i plus 1. This directly follows from the Neumann boundary condition of the left boundary. Well, the value to be inserted will be 1 over h square, ok. There will be no gamma and all that because you are taking the derivative with respect to u in order to find out the Jacobian. And it is really a matter of doing it on your own and seeing it.

So, now, if you are at the right boundary, so if i equal to info mx minus 1, then what will you do? Again, you have to insert only 4 columns. So, in cols will become equal to 4 and col 3 dot j will be equal to j, but this should be col; and col 3 dot i will be equal to i minus 1 because that is the inner boundary and the value will be simply 2 divided by h square, ok. So, far we have made what values you need to insert, how many values you need to insert.

(Refer Slide Time: 54:08)



```
            ncols = 4;
            col[3].j = j; col[3].i = i+1; v[3] = 2/(h*h);
          }

          if (i==info->max-1)
          {
            ncols = 4;
            col[3].j = j; col[3].i = i-1; v[3] = 2/(h*h);
          }

          MatSetValuesStencil(P, 1, &row, ncols, col, v, INSERT_VALUES);

        }
    }
    MatAssemblyBegin(P, MAT_FINAL_ASSEMBLY);
    MatAssemblyEnd(P, MAT_FINAL_ASSEMBLY);

    if (J!=P)
    {
      MatAssemblyBegin(J, MAT_FINAL_ASSEMBLY);
      MatAssemblyEnd(J, MAT_FINAL_ASSEMBLY);
    }


    return 0;
}


int main(int argc, char**argv)
```

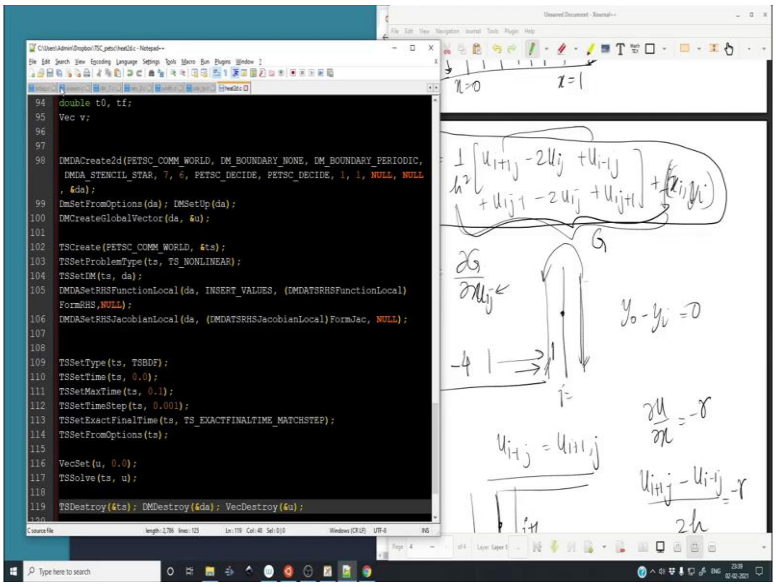
$x=0$
$x=1$
$\frac{1}{h^2}\left[u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1}\right] + f(x_i, y_i)$
$G$
$\frac{\partial G}{\partial u_{ij}}$
$-4$
$y_0 - y_i = 0$
$\frac{\partial u}{\partial x} = -\gamma$
$u_{i+1j} = u_{i+1,j}$
$\frac{u_{i+1j} - u_{i-1j}}{2h} = -\gamma$

Then we have to do MatSetValuesStencil. So, we have to pass the pre-conditioner, one row the address of the row, ncols number of columns and the col address, then the values and how do you want to insert them, so it is insert values, ok. So, now that you have done this we can come out of the loop. And so, where does the loop end?

Sorry, yeah, it is over here. So, this is the end of the loop. Once the loop ends we can write MatAssemblyBegin P comma MAT FINAL ASSEMBLY MatAssemblyEnd P comma MAT FINAL ASSEMBLY. So, this this is just how you define it. Finally, you have to set the Jacobian equal to P we have seen this in the previous lecture as well.

So, if J not equal to sorry not equal to P, then you have to simply do the assembly for J as well. Regardless of whether you have already done it for P you have to also do this, this is going to be simply J. So, finally, if everything is successful we return as 0. This is how you define the Jacobian function col back, that is form jack function. So, now we have found the Jacobian, we have done everything now what do we do? We must continue with the functions.

(Refer Slide Time: 56:16)



```
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);


TSSetType(ts, TSBDF);
TSSetTime(ts, 0.0);
TSSetMaxTime(ts, 0.1);
TSSetTimeStep(ts, 0.001);
TSSetExactFinalTime(ts, TS_EXACTFINALTIME_MATCHSTEP);
TSSetFromOptions(ts);

VecSet(u, 0.0);
TSSolve(ts, u);

TSDestroy(&ts); DMDestroy(&da); VecDestroy(&u);
```

$x=0$
$x=1$
$\frac{1}{h^2}\left[ u_{i+1j} - 2u_{ij} + u_{i-1j} + u_{ij-1} - 2u_{ij} + u_{ij+1} \right] + f(x_i, y_j)$
$G$
$\frac{\partial G}{\partial u_{ij}}$
$-4$
$y_0 - y_j = 0$
$\frac{\partial u}{\partial x} = -r$
$u_{i-1j} = u_{i+1,j}$
$\frac{u_{i+1j} - u_{i-1j}}{2h} = -r$

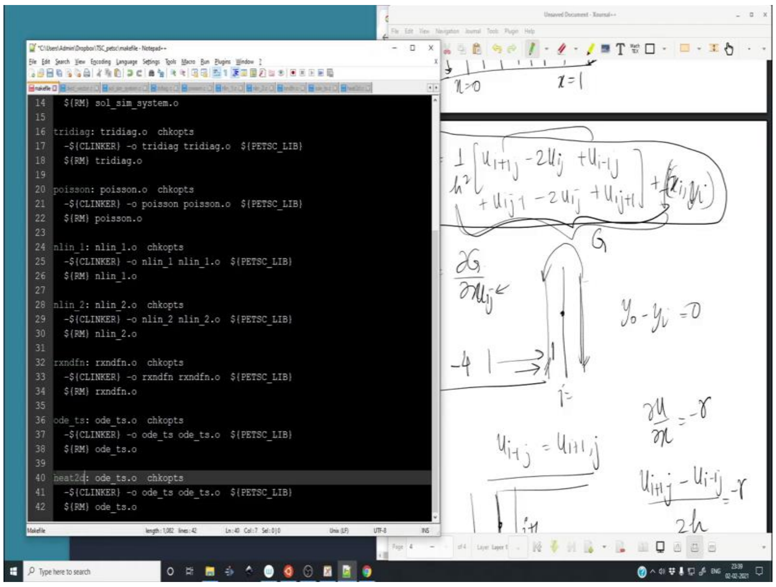
So, TSSetType ts. So, let us use a BDF solver a multistep solver TS BDF, then TSSetTime, TS this is the initial time TSSetMaxTime. We can set it to be 0.1; this is the time step. Now, this is the max time. Actually, max time is not the time step, the time step is set using TSSetTimeStep, TS this is the max time step, ok.

In fact, lastly we have to do TSSetExactFinalTime ts, TS EXACT FINAL TIME MATCH STEP. So, we must match the last time, so that it finishes at t equal to 0.1. Then after this we need to allow TS to be settable using the command argument. So, TSSetFromOptions ts, ok.

So, then what else do we need to do? Yeah. We can simply set all the values of the vector to be 0. So, VecSet u to be 0.0, then TSSolve ts comma u, finally, we must destroy everything. So, we have TSDestroy ts DMDestroy the address of ds this has to be the address of ts.

So, in order to destroy you must destroy the address, ok. Then, what else can we destroy? So, destroyed the TS, we have to destroy the vector as well. Let destroy the address of u is passed, ok. So, so far so good. So, let us create a new target for all this. So, make a new target.

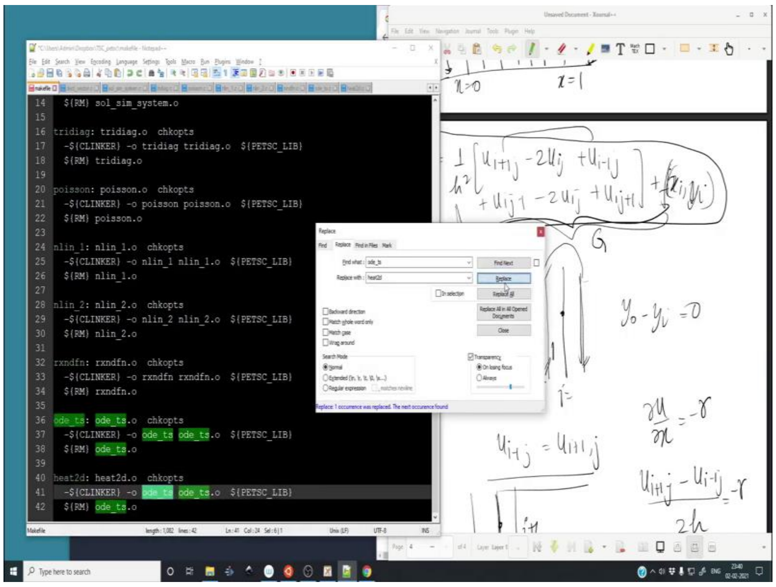
(Refer Slide Time: 58:50)



```
    ${RM} sol_sim_system.o

tridiag: tridiag.o  chkopts
    -${CLINKER} -o tridiag tridiag.o  ${PETSC_LIB}
    ${RM} tridiag.o

poisson: poisson.o  chkopts
    -${CLINKER} -o poisson poisson.o  ${PETSC_LIB}
    ${RM} poisson.o

nlin_1: nlin_1.o  chkopts
    -${CLINKER} -o nlin_1 nlin_1.o  ${PETSC_LIB}
    ${RM} nlin_1.o

nlin_2: nlin_2.o  chkopts
    -${CLINKER} -o nlin_2 nlin_2.o  ${PETSC_LIB}
    ${RM} nlin_2.o

rxndfn: rxndfn.o  chkopts
    -${CLINKER} -o rxndfn rxndfn.o  ${PETSC_LIB}
    ${RM} rxndfn.o

ode_ts: ode_ts.o  chkopts
    -${CLINKER} -o ode_ts ode_ts.o  ${PETSC_LIB}
    ${RM} ode_ts.o

heat2d: ode_ts.o  chkopts
    -${CLINKER} -o ode_ts ode_ts.o  ${PETSC_LIB}
    ${RM} ode_ts.o
```

So, target name will be heat2d.

(Refer Slide Time: 59:05)



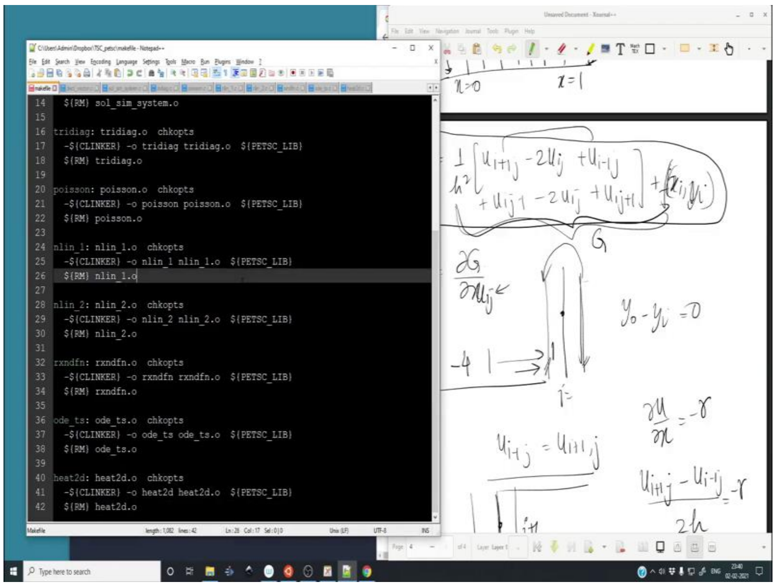
```
    ${RM} sol_sim_system.o

tridiag: tridiag.o  chkopts
    -${CLINKER} -o tridiag tridiag.o  ${PETSC_LIB}
    ${RM} tridiag.o

poisson: poisson.o  chkopts
    -${CLINKER} -o poisson poisson.o  ${PETSC_LIB}
    ${RM} poisson.o

nlin_1: nlin_1.o  chkopts
    -${CLINKER} -o nlin_1 nlin_1.o  ${PETSC_LIB}
    ${RM} nlin_1.o

nlin_2: nlin_2.o  chkopts
    -${CLINKER} -o nlin_2 nlin_2.o  ${PETSC_LIB}
    ${RM} nlin_2.o

rxndfn: rxndfn.o  chkopts
    -${CLINKER} -o rxndfn rxndfn.o  ${PETSC_LIB}
    ${RM} rxndfn.o

ode_ts: ode_ts.o  chkopts
    -${CLINKER} -o ode_ts ode_ts.o  ${PETSC_LIB}
    ${RM} ode_ts.o

heat2d: heat2d.o  chkopts
    -${CLINKER} -o ode_ts ode_ts.o  ${PETSC_LIB}
    ${RM} ode_ts.o
```

Find what : ode_ts
Replace with : heat2d

Replace: 1 occurence was replaced. The next occurence found

So, ode ts we must replace my heat 2d. Find next, oops; replace, replace, replace, replace, ok.

(Refer Slide Time: 59:49)



```
  ${RM} sol_sim_system.o

tridiag: tridiag.o  chkopts
  -${CLINKER} -o tridiag tridiag.o  ${PETSC_LIB}
  ${RM} tridiag.o

poisson: poisson.o  chkopts
  -${CLINKER} -o poisson poisson.o  ${PETSC_LIB}
  ${RM} poisson.o

nlin_1: nlin_1.o  chkopts
  -${CLINKER} -o nlin_1 nlin_1.o  ${PETSC_LIB}
  ${RM} nlin_1.o

nlin_2: nlin_2.o  chkopts
  -${CLINKER} -o nlin_2 nlin_2.o  ${PETSC_LIB}
  ${RM} nlin_2.o

rxndfn: rxndfn.o  chkopts
  -${CLINKER} -o rxndfn rxndfn.o  ${PETSC_LIB}
  ${RM} rxndfn.o

ode_ts: ode_ts.o  chkopts
  -${CLINKER} -o ode_ts ode_ts.o  ${PETSC_LIB}
  ${RM} ode_ts.o

heat2d: heat2d.o  chkopts
  -${CLINKER} -o heat2d heat2d.o  ${PETSC_LIB}
  ${RM} heat2d.o
```

$x=0$ $\quad x=1$

$$\underbrace{\frac{1}{h^2}\left[u_{i+1j}-2u_{ij}+u_{i-1j}+u_{ij-1}-2u_{ij}+u_{ij+1}\right]+f(x_i,y_i)}_{G}$$

$$\frac{\partial G}{\partial u_{ij}}$$

$-4 \quad 1$

$i=$

$y_0-y_i=0$

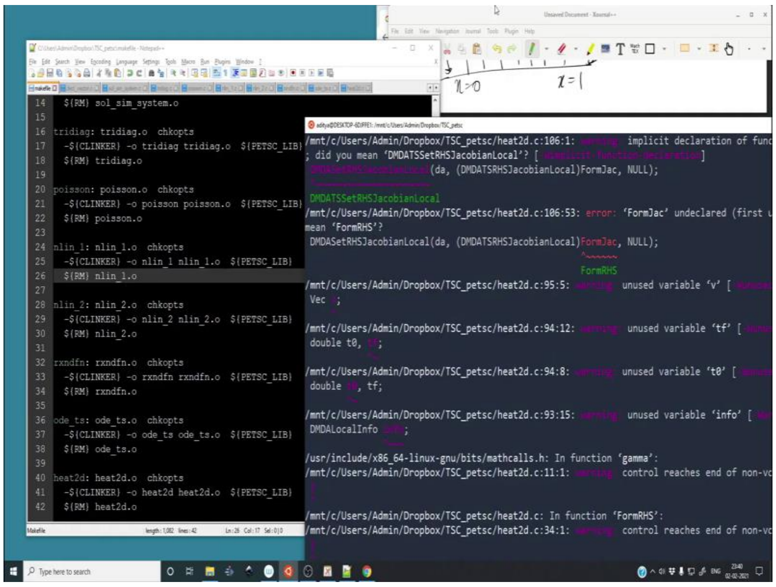
$\frac{\partial u}{\partial x}=-r$

$u_{i+j}=u_{i+1,j}$

$\frac{u_{i+1j}-u_{i-1j}}{2h}=-r$

$i+1$

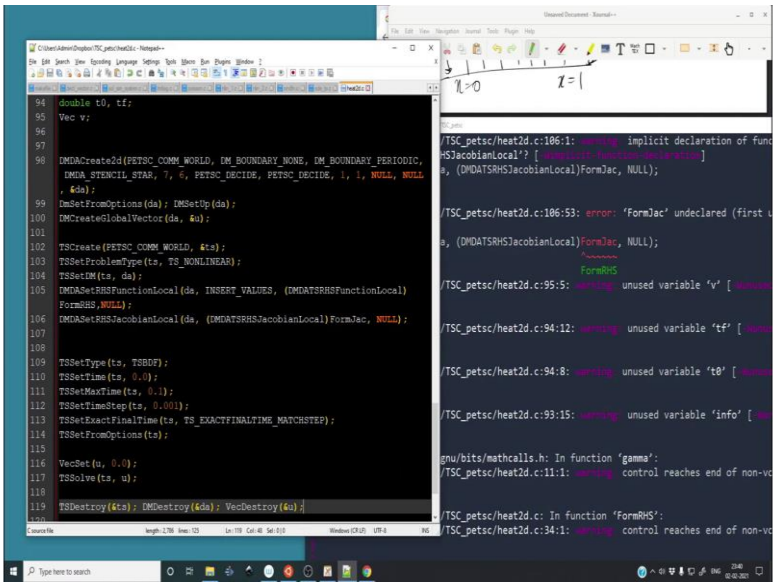
(Refer Slide Time: 59:51)



```
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:106:1: warning: implicit declaration of func
; did you mean 'DMDATSSetRHSJacobianLocal'? [-Wimplicit-function-declaration]
 DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);
 ^~~~~~~~~~~~~~~~~~~~~~~
 DMDATSSetRHSJacobianLocal
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:106:53: error: 'FormJac' undeclared (first u
mean 'FormRHS'?
 DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);
                                                     ^~~~~~~
                                                     FormRHS
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:95:5: warning: unused variable 'v' [
 Vec v;
     ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:12: warning: unused variable 'tf' [
 double t0, tf;
            ^~
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:8: warning: unused variable 't0' [
 double t0, tf;
        ^~
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:93:15: warning: unused variable 'info' [
 DMDALocalInfo info;
               ^~~~
/usr/include/x86_64-linux-gnu/bits/mathcalls.h: In function 'gamma':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:11:1: warning: control reaches end of non-vo
 }
 ^
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1: warning: control reaches end of non-vo
 }
 ^
```

So, let us make and see what errors we get. So, make heat2d, ok. So, let us see we appear to have a bunch of unused variables.

(Refer Slide Time: 60:08)

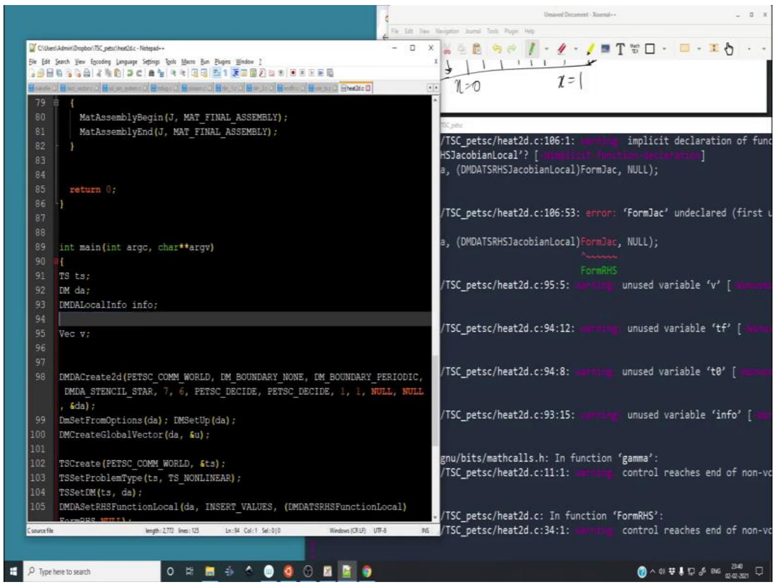


```c
double t0, tf;
Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDASetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);


TSSetType(ts, TSBDF);
TSSetTime(ts, 0.0);
TSSetMaxTime(ts, 0.1);
TSSetTimeStep(ts, 0.001);
TSSetExactFinalTime(ts, TS_EXACTFINALTIME_MATCHSTEP);
TSSetFromOptions(ts);

VecSet(u, 0.0);
TSSolve(ts, u);

TSDestroy(&ts); DMDestroy(&da); VecDestroy(&u);
```

```
/TSC_petsc/heat2d.c:106:1: warning: implicit declaration of func
HSJacobianLocal'? [-Wimplicit-function-declaration]
a, (DMDATSRHSJacobianLocal)FormJac, NULL);

/TSC_petsc/heat2d.c:106:53: error: 'FormJac' undeclared (first u

a, (DMDATSRHSJacobianLocal)FormJac, NULL);
                           ^~~~~~~
                           FormRHS
/TSC_petsc/heat2d.c:95:5: warning: unused variable 'v' [

/TSC_petsc/heat2d.c:94:12: warning: unused variable 'tf' [

/TSC_petsc/heat2d.c:94:8: warning: unused variable 't0' [

/TSC_petsc/heat2d.c:93:15: warning: unused variable 'info' [

gnu/bits/mathcalls.h: In function 'gamma':
/TSC_petsc/heat2d.c:11:1: warning: control reaches end of non-vo

/TSC_petsc/heat2d.c: In function 'FormRHS':
/TSC_petsc/heat2d.c:34:1: warning: control reaches end of non-vo
```

So, instead of setting the t 0 and t f we have hard core, anyway, so it does not matter. We can get rid of this.

(Refer Slide Time: 60:22)

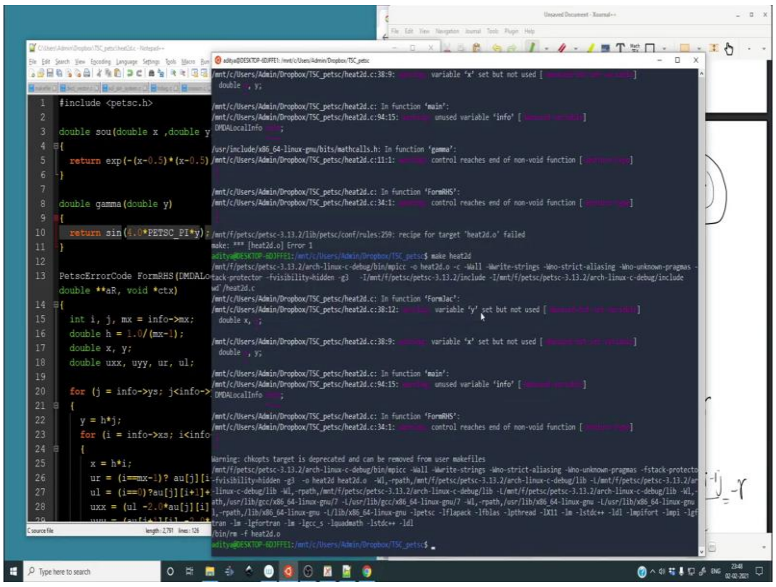


```c
  {
    MatAssemblyBegin(J, MAT_FINAL_ASSEMBLY);
    MatAssemblyEnd(J, MAT_FINAL_ASSEMBLY);
  }


  return 0;
}


int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;

Vec v;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DmSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDASetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
```

```
/TSC_petsc/heat2d.c:106:1: warning: implicit declaration of func
HSJacobianLocal'? [-Wimplicit-function-declaration]
a, (DMDATSRHSJacobianLocal)FormJac, NULL);

/TSC_petsc/heat2d.c:106:53: error: 'FormJac' undeclared (first u

a, (DMDATSRHSJacobianLocal)FormJac, NULL);
                           ^~~~~~~
                           FormRHS
/TSC_petsc/heat2d.c:95:5: warning: unused variable 'v' [

/TSC_petsc/heat2d.c:94:12: warning: unused variable 'tf' [

/TSC_petsc/heat2d.c:94:8: warning: unused variable 't0' [

/TSC_petsc/heat2d.c:93:15: warning: unused variable 'info' [

gnu/bits/mathcalls.h: In function 'gamma':
/TSC_petsc/heat2d.c:11:1: warning: control reaches end of non-vo

/TSC_petsc/heat2d.c: In function 'FormRHS':
/TSC_petsc/heat2d.c:34:1: warning: control reaches end of non-vo
```

So, that should be fine. Let me run it again, ok. FormJac is undeclared, why is that? We have clearly declared FormJac, FormJac, ok.

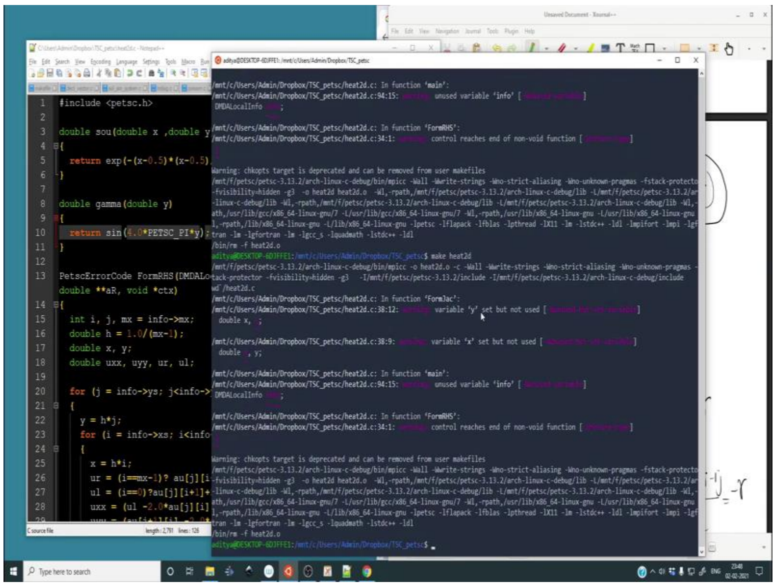
(Refer Slide Time: 61:14)



```
#include <petsc.h>

double sou(double x ,double y
{
  return exp(-(x-0.5)*(x-0.5)
}

double gamma(double y)
{
  return sin(4.0*PETSC_PI*y);
}

PetscErrorCode FormRHS(DMDALo
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1.0/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->
  {
    y = h*j;
    for (i = info->xs; i<info
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i
      ul = (i==0)?au[j][i+1]+
      uxx = (ul -2.0*au[j][i]
```

```
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:9:  variable 'x' set but not used [ ]
  double  , y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15:  unused variable 'info' [ ]
 DMDALocalInfo  ;

/usr/include/x86_64-linux-gnu/bits/mathcalls.h: In function 'gamma':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:11:1:  control reaches end of non-void function [ ]

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1:  control reaches end of non-void function [ ]

/mnt/f/petsc/petsc-3.13.2/lib/petsc/conf/rules:259: recipe for target 'heat2d.o' failed
make: *** [heat2d.o] Error 1
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make heat2d
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o heat2d.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
tack-protector -fvisibility=hidden -g3   -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include
wd`/heat2d.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormJac':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:12:  variable 'y' set but not used [ ]
  double x,  ;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:9:  variable 'x' set but not used [ ]
  double  , y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15:  unused variable 'info' [ ]
 DMDALocalInfo  ;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1:  control reaches end of non-void function [ ]

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ _
```

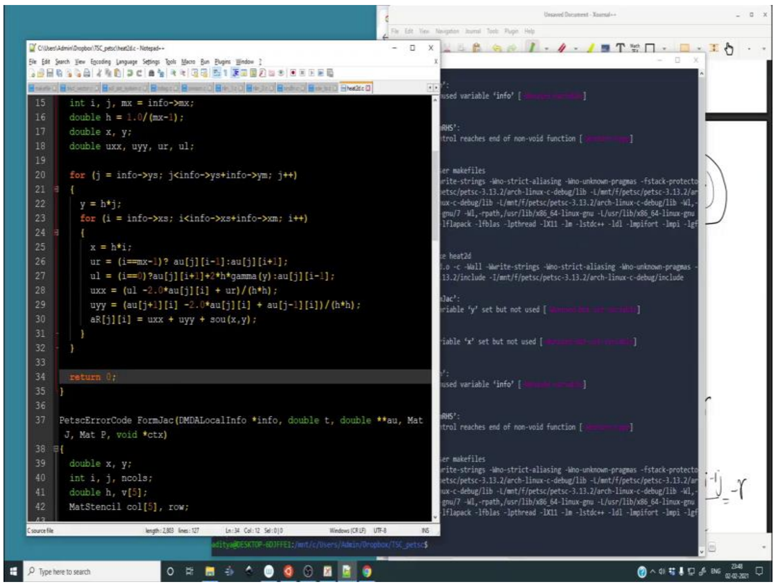
(Refer Slide Time: 61:15)



```
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15:  unused variable 'info' [ ]
 DMDALocalInfo  ;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1:  control reaches end of non-void function [ ]

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make heat2d
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o heat2d.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
tack-protector -fvisibility=hidden -g3   -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include
wd`/heat2d.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormJac':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:12:  variable 'y' set but not used [ ]
  double x,  ;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:9:  variable 'x' set but not used [ ]
  double  , y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15:  unused variable 'info' [ ]
 DMDALocalInfo  ;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1:  control reaches end of non-void function [ ]

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ _
```

So, the program compiles, but with the warning and the warning for, for mrhs, it does not have a return value. Let us have a look whether it has a written value. No, it does not. So, we must return 0, if everything works properly. So, it has written 0.

(Refer Slide Time: 61:28)

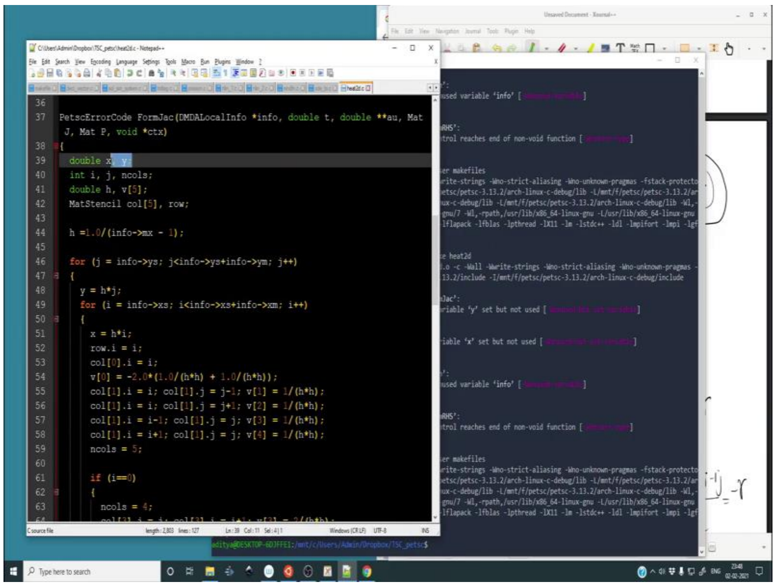


```
15   int i, j, mx = info->mx;
16   double h = 1.0/(mx-1);
17   double x, y;
18   double uxx, uyy, ur, ul;
19
20   for (j = info->ys; j<info->ys+info->ym; j++)
21   {
22     y = h*j;
23     for (i = info->xs; i<info->xs+info->xm; i++)
24     {
25       x = h*i;
26       ur = (i==mx-1)? au[j][i-1]:au[j][i+1];
27       ul = (i==0)?au[j][i+1]+2*h*gamma(y):au[j][i-1];
28       uxx = (ul -2.0*au[j][i] + ur)/(h*h);
29       uyy = (au[j+1][i] -2.0*au[j][i] + au[j-1][i])/(h*h);
30       aR[j][i] = uxx + uyy + sou(x,y);
31     }
32   }
33
34   return 0;
35 }
36
37 PetscErrorCode FormJac(DMDALocalInfo *info, double t, double **au, Mat
    J, Mat P, void *ctx)
38 {
39   double x, y;
40   int i, j, ncols;
41   double h, v[5];
42   MatStencil col[5], row;
```

```
':
used variable 'info' [[illegible]]

RHS':
trol reaches end of non-void function [[illegible]]

er makefiles
rite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protecto
etsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
ux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf

e heat2d
.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include

Jac':
riable 'y' set but not used [[illegible]]

iable 'x' set but not used [[illegible]]

':
used variable 'info' [[illegible]]

RHS':
trol reaches end of non-void function [[illegible]]

er makefiles
rite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protecto
etsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
ux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf

aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

What else we have not used? x and y in the main, no, no it is not in the main, line 38.

(Refer Slide Time: 61:43)

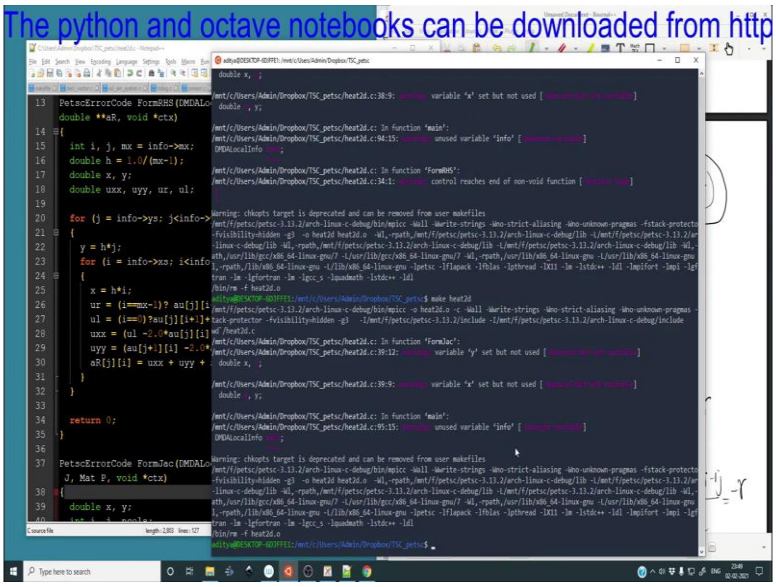


```
36
37 PetscErrorCode FormJac(DMDALocalInfo *info, double t, double **au, Mat
    J, Mat P, void *ctx)
38 {
39   double x, y;
40   int i, j, ncols;
41   double h, v[5];
42   MatStencil col[5], row;
43
44   h =1.0/(info->mx - 1);
45
46   for (j = info->ys; j<info->ys+info->ym; j++)
47   {
48     y = h*j;
49     for (i = info->xs; i<info->xs+info->xm; i++)
50     {
51       x = h*i;
52       row.i = i;
53       col[0].i = i;
54       v[0] = -2.0*(1.0/(h*h) + 1.0/(h*h));
55       col[1].i = i; col[1].j = j-1; v[1] = 1/(h*h);
56       col[1].i = i; col[1].j = j+1; v[2] = 1/(h*h);
57       col[1].i = i-1; col[1].j = j; v[3] = 1/(h*h);
58       col[1].i = i+1; col[1].j = j; v[4] = 1/(h*h);
59       ncols = 5;
60
61       if (i==0)
62       {
63         ncols = 4;
```

```
':
used variable 'info' [[illegible]]

RHS':
trol reaches end of non-void function [[illegible]]

er makefiles
rite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protecto
etsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
ux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf

e heat2d
.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include

Jac':
riable 'y' set but not used [[illegible]]

iable 'x' set but not used [[illegible]]

':
used variable 'info' [[illegible]]

RHS':
trol reaches end of non-void function [[illegible]]

er makefiles
rite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protecto
etsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
ux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf

aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

Double x comma y, but we have used x comma y;, it is said, but it is not used, ok, no problem. It is said, but it is not used. Not a big deal. 94 we have not used info that is ok.

(Refer Slide Time: 62:12)

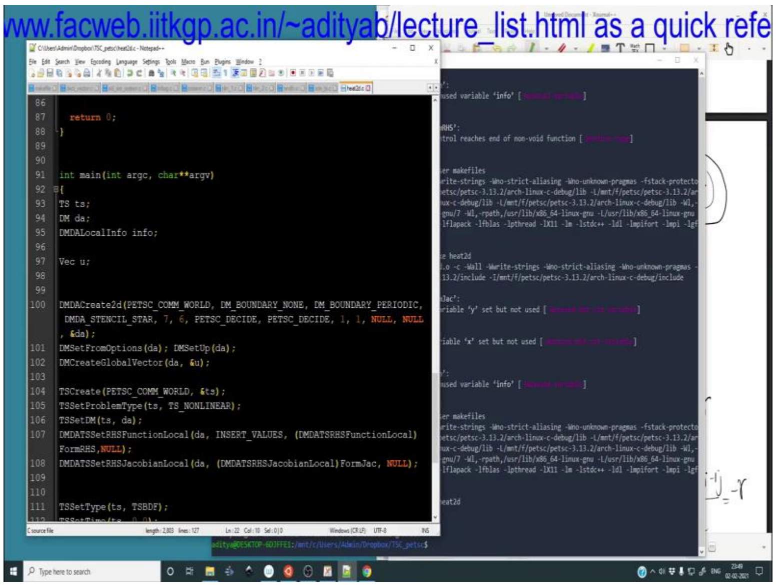

The python and octave notebooks can be downloaded from http

```
PetscErrorCode FormRHS(DMDALo
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1.0/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->
  {
    y = h*j;
    for (i = info->xs; i<info
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i
      ul = (i==0)?au[j][i+1]+
      uxx = (ul -2.0*au[j][i]
      uyy = (au[j+1][i] -2.0*
      aR[j][i] = uxx + uyy +
    }
  }

  return 0;
}

PetscErrorCode FormJac(DMDALo
 J, Mat P, void *ctx)
{
  double x, y;
```

```
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:38:9: warning: variable 'x' set but not used [-Wunused-but-set-variable]
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15: warning: unused variable 'info' [-Wunused-variable]
 DMDALocalInfo info;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1: warning: control reaches end of non-void function [-Wreturn-type]
 }

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make heat2d
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o heat2d.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
tack-protector -fvisibility=hidden -g3   -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include
wd`/heat2d.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormJac':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:39:12: warning: variable 'y' set but not used [-Wunused-but-set-variable]
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:39:9: warning: variable 'x' set but not used [-Wunused-but-set-variable]
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:95:15: warning: unused variable 'info' [-Wunused-variable]
 DMDALocalInfo info;

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

For mrhs, I thought we fixed that it reaches the end of a non-void function; well, ok. So, now, only a few warnings remain, but that is ok.

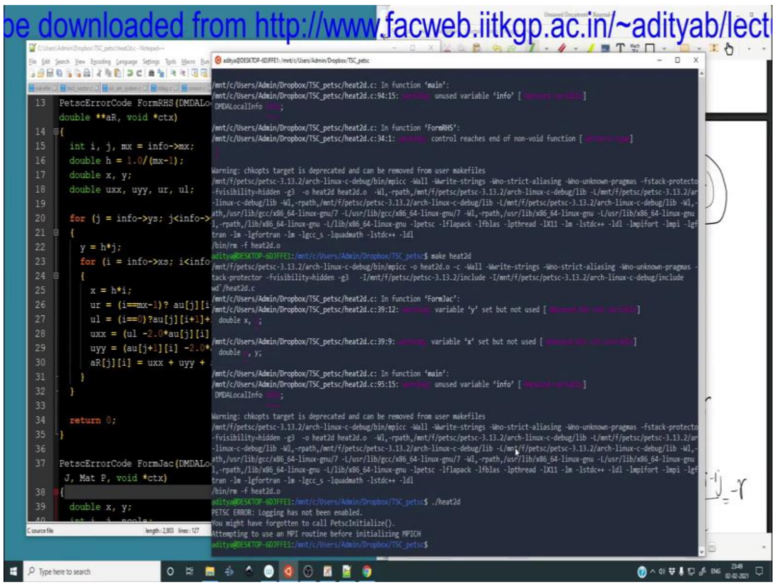
(Refer Slide Time: 62:30)


www.facweb.iitkgp.ac.in/~adityab/lecture_list.html as a quick refe

```
  return 0;
}


int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;

Vec u;


DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DMSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDATSSetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDATSSetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);


TSSetType(ts, TSBDF);
```

```
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

(Refer Slide Time: 62:34)

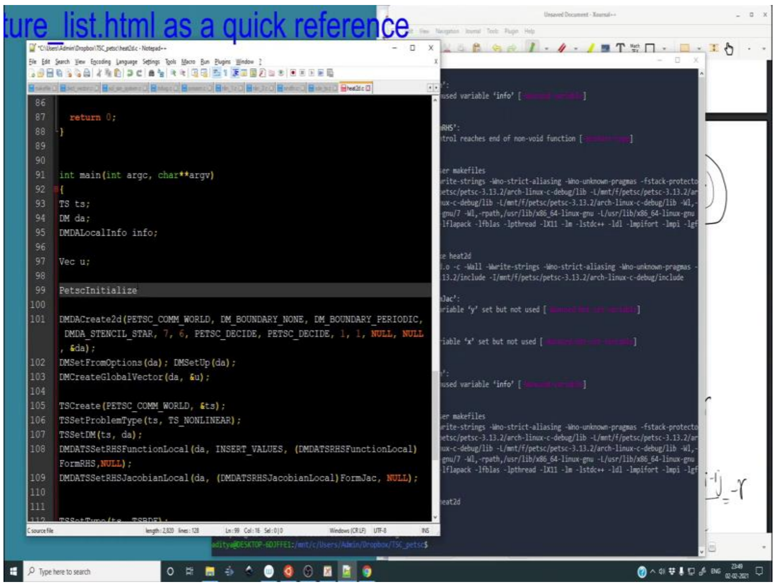

be downloaded from http://www.facweb.iitkgp.ac.in/~adityab/lect

```
PetscErrorCode FormRHS(DMDALo
double **aR, void *ctx)
{
  int i, j, mx = info->mx;
  double h = 1.0/(mx-1);
  double x, y;
  double uxx, uyy, ur, ul;

  for (j = info->ys; j<info->
  {
    y = h*j;
    for (i = info->xs; i<info
    {
      x = h*i;
      ur = (i==mx-1)? au[j][i
      ul = (i==0)?au[j][i+1]+
      uxx = (ul -2.0*au[j][i]
      uyy = (au[j+1][i] -2.0*
      aR[j][i] = uxx + uyy +
    }
  }

  return 0;
}

PetscErrorCode FormJac(DMDALo
 J, Mat P, void *ctx)
{
  double x, y;
```

```
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:94:15: warning: unused variable 'info' [-Wunused-variable]
 DMDALocalInfo info;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormRHS':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:34:1: warning: control reaches end of non-void function [-Wreturn-type]

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make heat2d
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o heat2d.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
tack-protector -fvisibility=hidden -g3   -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include
wd`/heat2d.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'FormJac':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:39:12: warning: variable 'y' set but not used [-Wunused-but-set-variable]
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:39:9: warning: variable 'x' set but not used [-Wunused-but-set-variable]
 double x, y;

/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c: In function 'main':
/mnt/c/Users/Admin/Dropbox/TSC_petsc/heat2d.c:95:15: warning: unused variable 'info' [-Wunused-variable]
 DMDALocalInfo info;

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-unknown-pragmas -fstack-protecto
-fvisibility=hidden -g3  -o heat2d heat2d.o  -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/ar
-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-
ath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu
1,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgf
tran -lm -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f heat2d.o
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./heat2d
PETSC ERROR: Logging has not been enabled.
You might have forgotten to call PetscInitialize().
Attempting to use an MPI routine before initializing MPICH
aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, now, we are in a position to run this particular program. So, dot its dot slash heat2d; I have forgotten to col petsc initialize.

(Refer Slide Time: 62:55)



```

  return 0;
}


int main(int argc, char**argv)
{
TS ts;
DM da;
DMDALocalInfo info;

Vec u;

PetscInitialize

DMDACreate2d(PETSC_COMM_WORLD, DM_BOUNDARY_NONE, DM_BOUNDARY_PERIODIC,
 DMDA_STENCIL_STAR, 7, 6, PETSC_DECIDE, PETSC_DECIDE, 1, 1, NULL, NULL
, &da);
DMSetFromOptions(da); DMSetUp(da);
DMCreateGlobalVector(da, &u);

TSCreate(PETSC_COMM_WORLD, &ts);
TSSetProblemType(ts, TS_NONLINEAR);
TSSetDM(ts, da);
DMDATSSetRHSFunctionLocal(da, INSERT_VALUES, (DMDATSRHSFunctionLocal)
FormRHS,NULL);
DMDATSSetRHSJacobianLocal(da, (DMDATSRHSJacobianLocal)FormJac, NULL);
```

Well, petsc is kind enough to inform you that you have forgotten to do that, and yeah such things do happen when you are pressed for time, and that is why I am wait; I thought we did initialize it somewhere. I recall initializing it somewhere. So, with this, we in this particular lecture. It is become too long. You can have a look at the program from the website and you can try to run it on your own. So, with this, it is goodbye from me until next time. Have a good day. Bye.