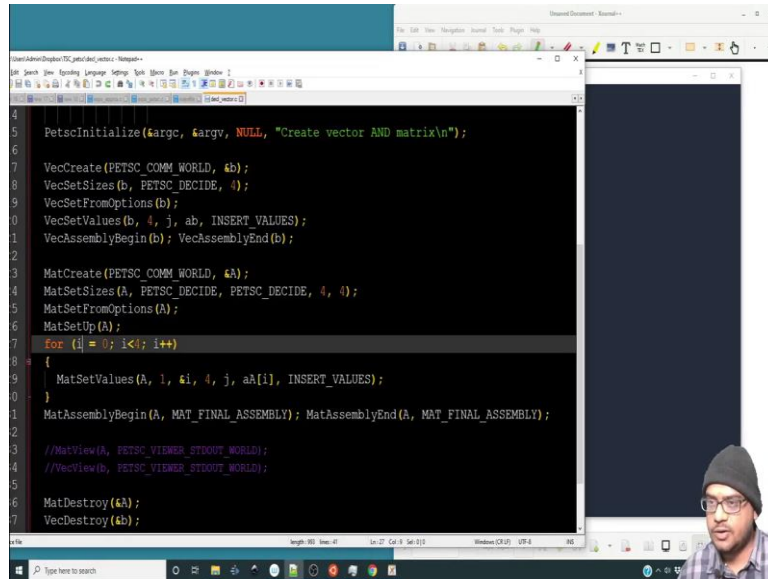


Tools in Scientific Computing
Prof. Aditya Bandopadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 31
KSP object and solving a system

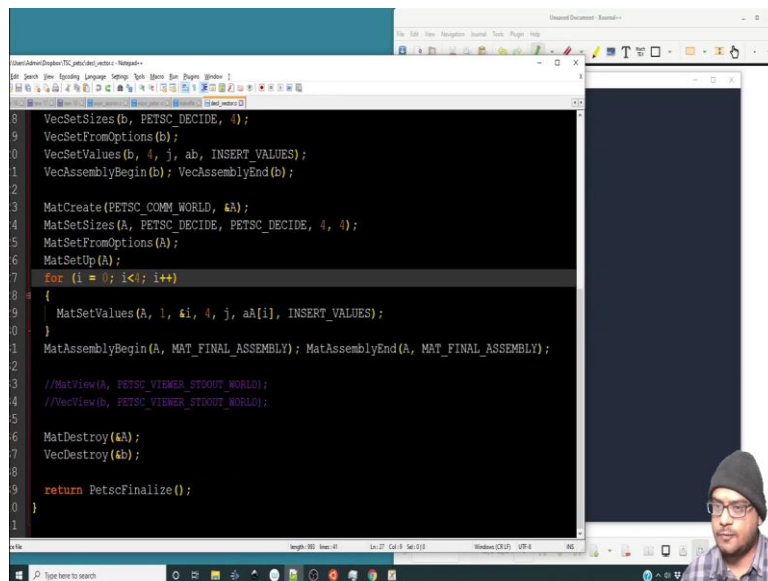
(Refer Slide Time: 00:27)



```
4 PetscInitialize(&argc, &argv, NULL, "Create vector and matrix\n");
5
6
7 VecCreate(PETSC_COMM_WORLD, &b);
8 VecSetSizes(b, PETSC_DECIDE, 4);
9 VecSetFromOptions(b);
10 VecSetValues(b, 4, j, ab, INSERT_VALUES);
11 VecAssemblyBegin(b); VecAssemblyEnd(b);
12
13 MatCreate(PETSC_COMM_WORLD, &A);
14 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
15 MatSetFromOptions(A);
16 MatSetUp(A);
17 for (i = 0; i < 4; i++)
18 {
19     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
20 }
21 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
22
23 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
24 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
25
26 MatDestroy(&A);
27 VecDestroy(&b);
```

Last class, we saw how to create a vector and a matrix.

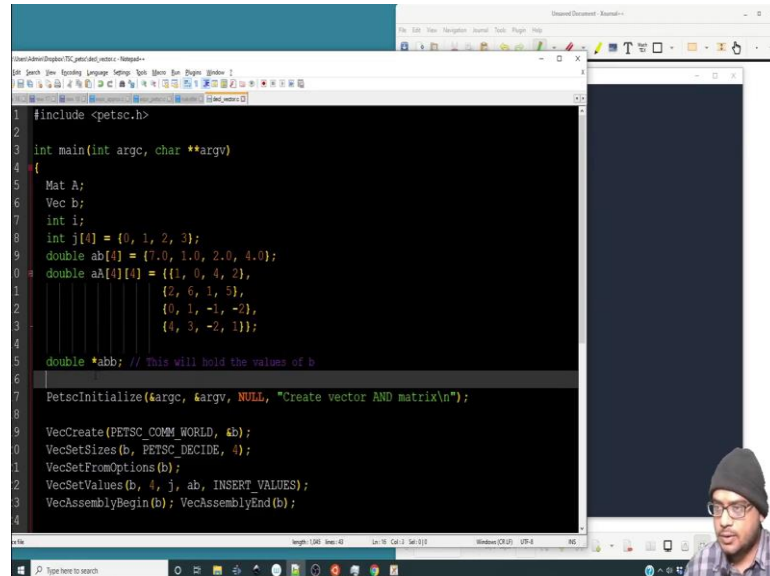
(Refer Slide Time: 00:33)



```
8 VecSetSizes(b, PETSC_DECIDE, 4);
9 VecSetFromOptions(b);
10 VecSetValues(b, 4, j, ab, INSERT_VALUES);
11 VecAssemblyBegin(b); VecAssemblyEnd(b);
12
13 MatCreate(PETSC_COMM_WORLD, &A);
14 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
15 MatSetFromOptions(A);
16 MatSetUp(A);
17 for (i = 0; i < 4; i++)
18 {
19     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
20 }
21 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
22
23 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
24 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
25
26 MatDestroy(&A);
27 VecDestroy(&b);
28
29 return PetscFinalize();
30 }
```

And in this particular video we are going to start off by looking at how we can print values to a command line and effectively to a file as well.

(Refer Slide Time: 00:51)

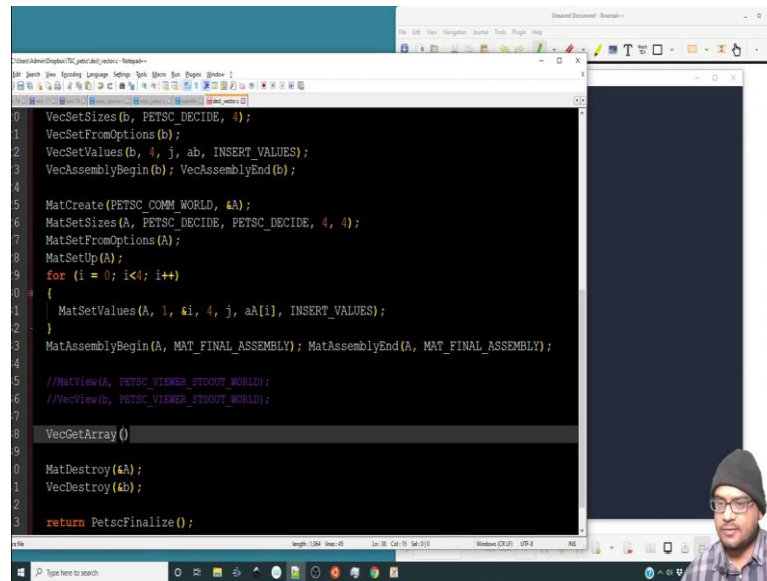


```
1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b;
7     int i;
8     int j[4] = {0, 1, 2, 3};
9     double ab[4] = {7.0, 1.0, 2.0, 4.0};
10    double aA[4][4] = {{1, 0, 4, 2},
11                      {2, 6, 1, 5},
12                      {0, 1, -1, -2},
13                      {4, 3, -2, 1}};
14
15    double *abb; // This will hold the values of b
16
17    PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
18
19    VecCreate(PETSC_COMM_WORLD, &b);
20    VecSetSizes(b, PETSC_DECIDE, 4);
21    VecSetFromOptions(b);
22    VecSetValues(b, 4, j, ab, INSERT_VALUES);
23    VecAssemblyBegin(b); VecAssemblyEnd(b);
24 }
```

So, for that we have to first create an array. So, suppose we want to print the vector. So, suppose we want to print the vector like this I mean of course, you can simply loop over the elements of ab, but that is not useful in this particular context because, once you start programmatically declaring your arrays you would not be able to have such an easy access.

So, what you will have is the Vec object b and eventually you want to print the elements of b alright. So, let us create an array. So, double star abb. So, I am using abb because ab is already used. So, essentially this holds rather this will hold the value of b alright. So, we have declared double abb. So, before destroying we have to write certain lines.

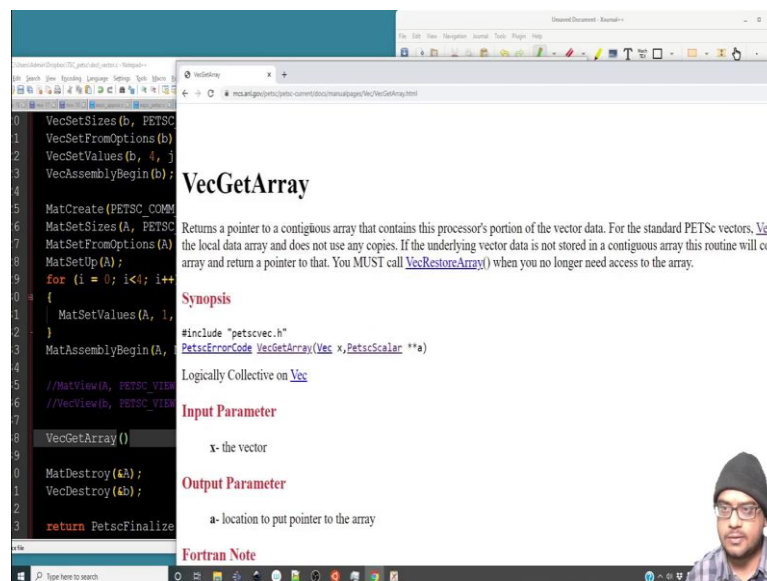
(Refer Slide Time: 02:07)



```
0 VecSetSizes(b, PETSC_DECIDE, 4);
1 VecSetFromOptions(b);
2 VecSetValues(b, 4, j, ab, INSERT_VALUES);
3 VecAssemblyBegin(b); VecAssemblyEnd(b);
4
5 MatCreate(PETSC_COMM_WORLD, &A);
6 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 For (i = 0; i<4; i++)
10 {
11   MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 VecGetArray ()
19
20 MatDestroy (&A);
21 VecDestroy (&b);
22
23 return PetscFinalize();
```

So, we will do VecGetArray. So, VecGetArray is a function.

(Refer Slide Time: 02:20)



VecGetArray

Returns a pointer to a contiguous array that contains this processor's portion of the vector data. For the standard PETSc vectors, `Vec` the local data array and does not use any copies. If the underlying vector data is not stored in a contiguous array this routine will copy array and return a pointer to that. You MUST call `VecRestoreArray()` when you no longer need access to the array.

Synopsis

```
#include "petscvec.h"
PetscErrorCode VecGetArray(Vec x, PetscScalar **a)
```

Logically Collective on `Vec`

Input Parameter

- x- the vector

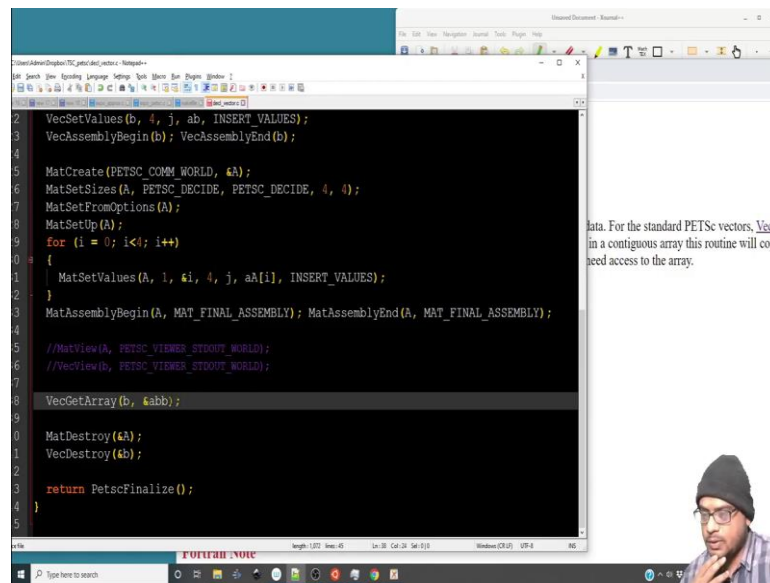
Output Parameter

- a- location to put pointer to the array

Fortran Note

So, let me just take you to the functional reference. So, VecGetArray returns an array which contains the processors portion of the vector data ok. So, VecGetArray requires the vector object x and the Petsc double star a. So, because we have declared it as this we have to pass the address of abb rather than simply passing abb.

(Refer Slide Time: 02:43)

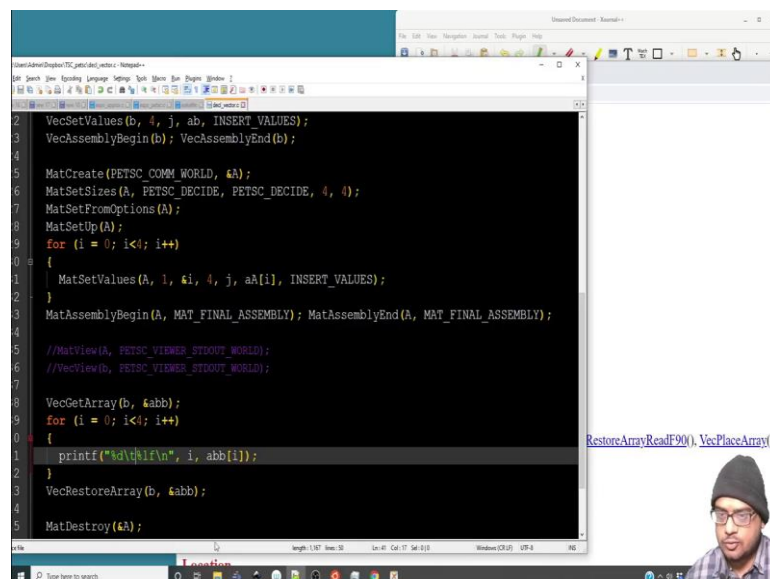


```
2 VecSetValues(b, 4, j, ab, INSERT_VALUES);
3 VecAssemblyBegin(b); VecAssemblyEnd(b);
4
5 MatCreate(PETSC_COMM_WORLD, &A);
6 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i<4; i++)
10 {
11     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 VecGetArray(b, &abb);
19
20 MatDestroy(&A);
21 VecDestroy(&b);
22
23 return PetscFinalize();
24 }
```

data. For the standard PETSc vectors, `VecG` in a contiguous array this routine will copy need access to the array.

So, we must pass b and the address of abb. So, another useful thing is You MUST call VecRestoreArray when you no longer need access to the array. So, PETSc tries to keep everything in it is own data structure by calling this your actually transferring the vec object to a simple c array abb. So, we must also follow up by the VecRestoreArray it has the same syntax.

(Refer Slide Time: 03:19)



```
2 VecSetValues(b, 4, j, ab, INSERT_VALUES);
3 VecAssemblyBegin(b); VecAssemblyEnd(b);
4
5 MatCreate(PETSC_COMM_WORLD, &A);
6 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i<4; i++)
10 {
11     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 VecGetArray(b, &abb);
19 for (i = 0; i<4; i++)
20 {
21     printf("%d\t%d\t%d\n", i, abb[i]);
22 }
23 VecRestoreArray(b, &abb);
24
25 MatDestroy(&A);
```

`RestoreArrayReadF90(), VecPlaceArray()`

So, let us do that. So, it is simply restore ok. So, this will now help us to print the values of abb. So, what we will do is for $i = 0, i < 4, i++$ `printf "%lf \n` and we will print abb i. In fact, let us print i as well. So, let us just put a `% d` like `\ t` alright.

(Refer Slide Time: 04:06)

```

1  VecSetValues(b, 4, j, ab, INSERT);
2  VecAssemblyBegin(b); VecAssemblyEnd(b);
3  MatCreate(PETSC_COMM_WORLD, &A);
4  MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, PETSC_DECIDE, PETSC_DECIDE);
5  MatSetFromOptions(A);
6  MatSetOp(A);
7  for (i = 0; i < 4; i++)
8  {
9      MatSetValues(A, 1, &i, j, aA, INSERT_VALUES);
10     MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
11     MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
12     VecGetArray(b, &abb);
13     for (i = 0; i < 4; i++)
14     {
15         printf("%d\t%lf\n", i, abb[i]);
16     }
17     VecRestoreArray(b, &abb);
18 }
19 MatDestroy(&A);

```

So, let us compile this. So, the make file will be the make target will be make declare vector alright dot slash declare vector ok. So, we could finally print this and at the end of restore it d allocates abb. So, once this line is executed we cannot do this anymore.

(Refer Slide Time: 04:41)

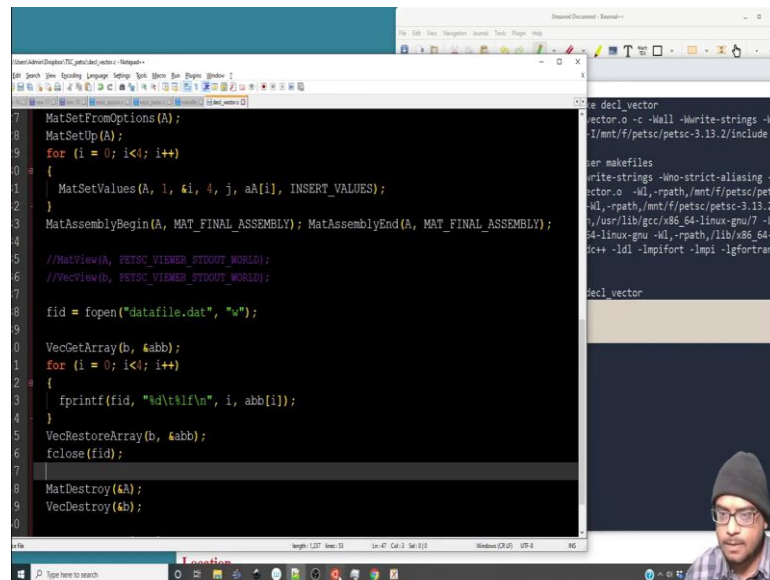
```

1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b;
7     int i;
8     int j[4] = {0, 1, 2, 3};
9     double ab[4] = {7.0, 1.0, 2.0, 4.0};
10    double aA[4][4] = {{1, 0, 4, 2},
11                      {2, 6, 1, 5},
12                      {0, 1, -1, -2},
13                      {4, 3, -2, 1}};
14
15    double *abb; // This will hold the values of b
16    FILE *fid;
17    PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
18
19    VecCreate(PETSC_COMM_WORLD, &b);
20    VecSetSizes(b, PETSC_DECIDE, 4);
21    VecSetFromOptions(b);
22    VecSetValues(b, 4, j, ab, INSERT_VALUES);
23    VecAssemblyBegin(b); VecAssemblyEnd(b);

```

So, now you can imagine instead of having a print f you could alternately do an fprintf. So, there is an fid and then you simply fclose fid. So, for that we have to declare FILE star fid then.

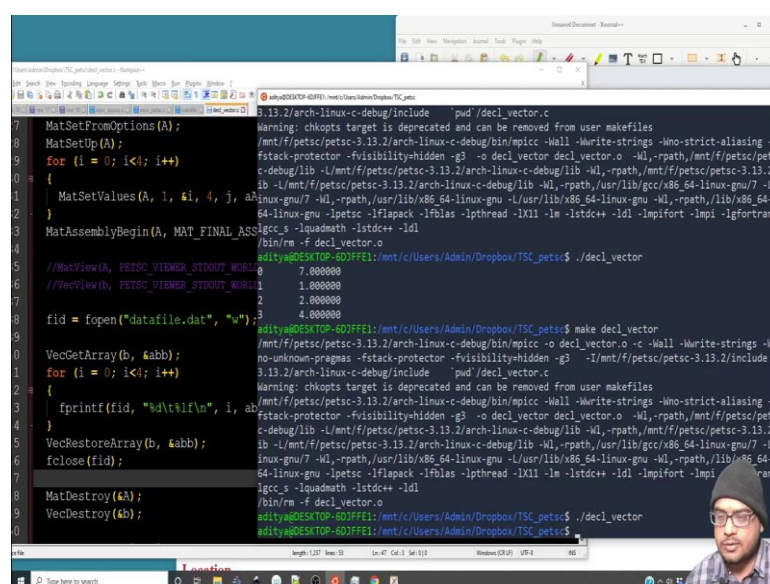
(Refer Slide Time: 05:03)



```
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i<I; i++)
10 {
11     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 fid = fopen("datafile.dat", "w");
19
20 VecGetArray(b, &abb);
21 for (i = 0; i<I; i++)
22 {
23     fprintf(fid, "%d\t%f\n", i, abb[i]);
24 }
25 VecRestoreArray(b, &abb);
26 fclose(fid);
27
28 MatDestroy(&A);
29 VecDestroy(&b);
```

So, it is a pointer to a file then fid equal to fopen datafile dot dat right and yeah that is pretty much it. So, this should allow us to write the data to a file and then close the file as well let us see; let us see what happens.

(Refer Slide Time: 05:27)



```
3.13.2/arch-linux-c-debug/include -pud /decl_vector.c
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -W
fstack-protector -fvvisibility-hidden -g3 -o decl_vector decl_vector.o -Wl,-rpath,/mnt/ff/petsc/pets
c-3.13.2/arch-linux-c-debug/lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/
-L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/
usr/lib/gcc/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-
64-linux-gnu -lpetsc -lflapack -lfflas -lpthread -lx11 -lm -lstdc++ -ldl -lmpifort -lmpi -lfortran
-lquadmath -lstdc++ -ldl
/bin/rm -f decl_vector.o
aditya@DESKTOP-6D3FFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./decl_vector
0 7.000000
1 1.000000
2 2.000000
3 4.000000
aditya@DESKTOP-6D3FFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make decl_vector
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o decl_vector.o -c -Wall -Wwrite-strings -Wno
no-unknown-pragmas -fstack-protector -fvvisibility-hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include
3.13.2/arch-linux-c-debug/include -pud /decl_vector.c
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -W
fstack-protector -fvvisibility-hidden -g3 -o decl_vector decl_vector.o -Wl,-rpath,/mnt/ff/petsc/pets
c-3.13.2/arch-linux-c-debug/lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/
-L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/
linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-
64-linux-gnu -lpetsc -lflapack -lfflas -lpthread -lx11 -lm -lstdc++ -ldl -lmpifort -lmpi -lfortran
-lquadmath -lstdc++ -ldl
/bin/rm -f decl_vector.o
aditya@DESKTOP-6D3FFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./decl_vector
aditya@DESKTOP-6D3FFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

(Refer Slide Time: 05:35)

```
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i < 4; i++)
10 {
11     MatSetValues(A, 1, i, 4, j, a);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17 for (i = 0; i < 4; i++)
18 {
19     fprintf(fid, "%d\t%f\n", i, a);
20 }
21 VecRestoreArray(b, &abb);
22 fclose(fid);
23 MatDestroy(&A);
24 VecDestroy(&b);
```

So, let us go to the folder and see whether we have something datafile dot dat, let me why is it not opening.

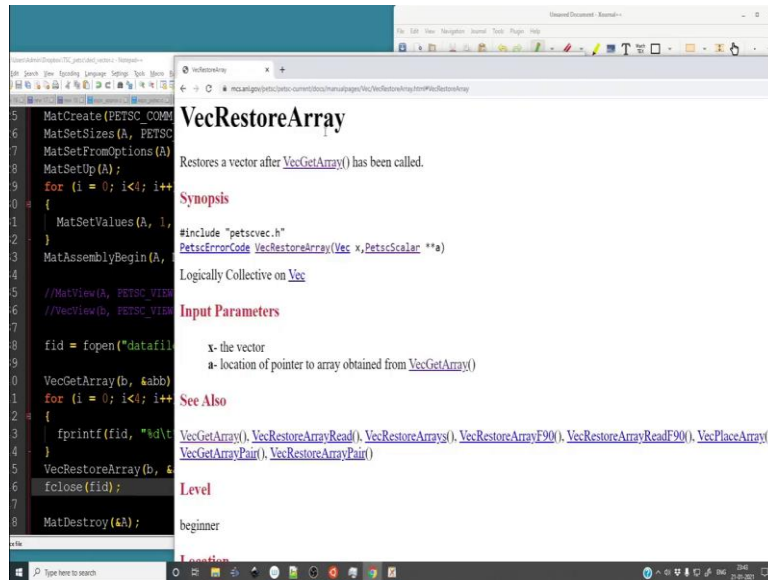
(Refer Slide Time: 05:46)

```
1 0 7.000000
2 1 1.000000
3 2 2.000000
4 3 4.000000
```

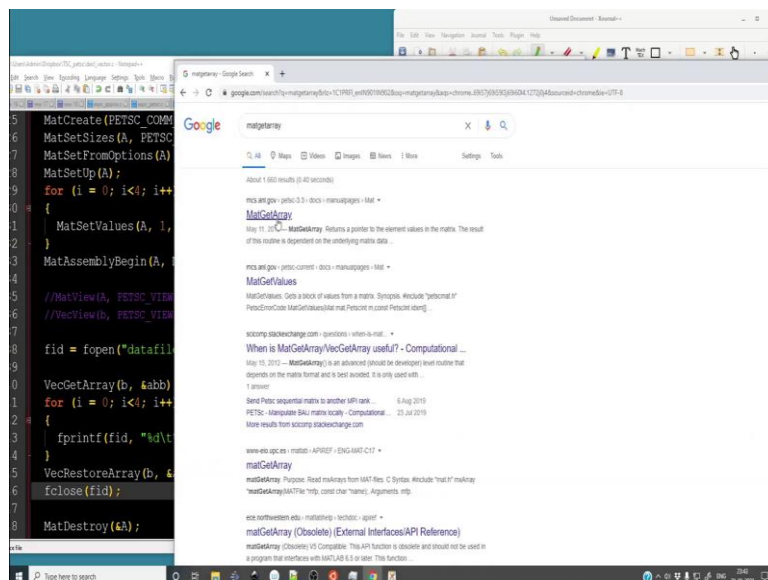
Great. So, we do have the data inside the database. So, it is quite useful when you want to. So, suppose you want to post process data once you have this you can then plot it using Python or glue plot anything you like it is always useful to have the data file when you are working with it.

So, this is a small snippet and I am not going to discuss more about this we will use it when we will need to ok. So, it is quite useful a versatile bit of snippet which you will find useful time to time.

(Refer Slide Time: 06:31)

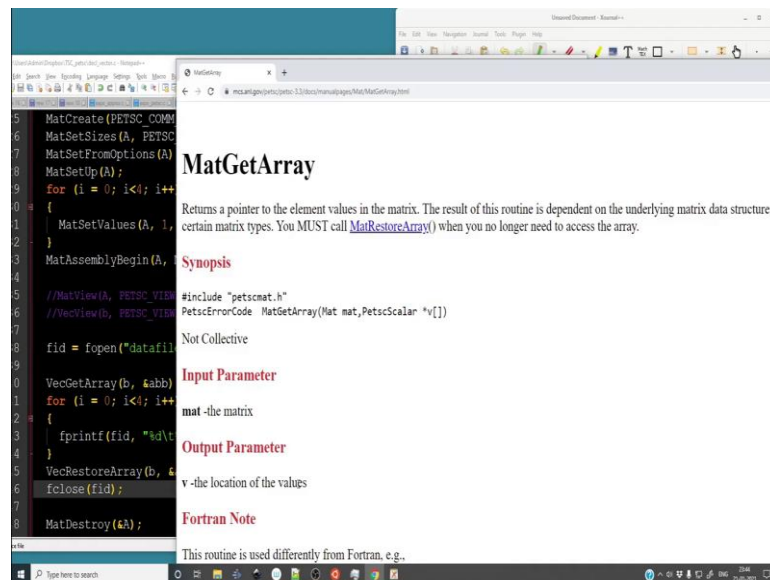


(Refer Slide Time: 06:33)



So, similarly similar to the `VecGetArray` there is also `mat get` rather `MatGetArray`.

(Refer Slide Time: 06:37)



```
5 MatCreate(PETSC_COMM_WORLD, &A, &B, &C);
6 MatSetSizes(A, PETSC_DEFAULT, PETSC_DEFAULT, &A);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i <= N; i++)
10 {
11     MatSetValues(A, 1, &A, &B, &C);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 fid = fopen("datafile.txt", "w");
19
20 VecGetArray(b, &abb);
21 for (i = 0; i <= N; i++)
22 {
23     fprintf(fid, "%d\n", *abb);
24 }
25 VecRestoreArray(b, &abb);
26 fclose(fid);
27
28 MatDestroy(&A);
```

MatGetArray

Returns a pointer to the element values in the matrix. The result of this routine is dependent on the underlying matrix data structure, and certain matrix types. You MUST call [MatRestoreArray\(\)](#) when you no longer need to access the array.

Synopsis

```
#include "petscmat.h"
PetscErrorCode MatGetArray(Mat mat, PetscScalar *v[])
```

Not Collective

Input Parameter

mat - the matrix

Output Parameter

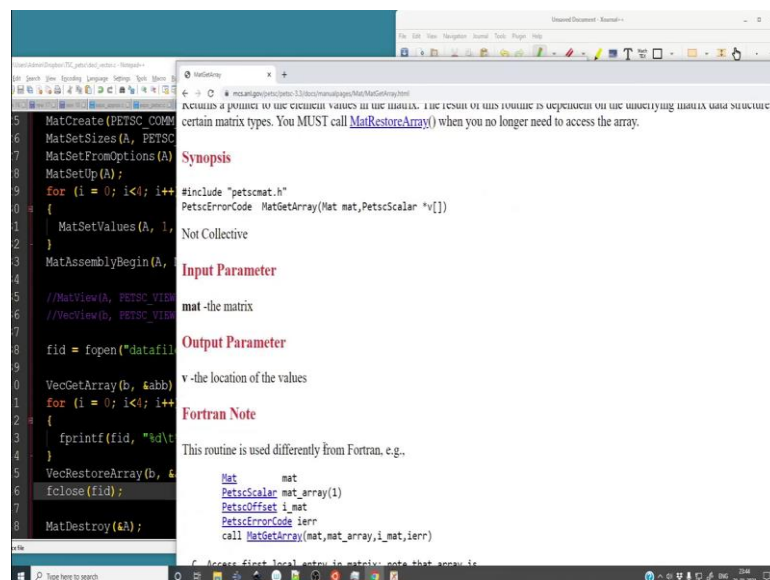
v - the location of the values

Fortran Note

This routine is used differently from Fortran, e.g.,

So, it does the same thing right instead it has to be passed with Mat and PetscScalar the pointer to v ok the location of the values.

(Refer Slide Time: 06:48)



```
5 MatCreate(PETSC_COMM_WORLD, &A, &B, &C);
6 MatSetSizes(A, PETSC_DEFAULT, PETSC_DEFAULT, &A);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i <= N; i++)
10 {
11     MatSetValues(A, 1, &A, &B, &C);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17
18 fid = fopen("datafile.txt", "w");
19
20 VecGetArray(b, &abb);
21 for (i = 0; i <= N; i++)
22 {
23     fprintf(fid, "%d\n", *abb);
24 }
25 VecRestoreArray(b, &abb);
26 fclose(fid);
27
28 MatDestroy(&A);
```

MatGetArray

Returns a pointer to the element values in the matrix. The result of this routine is dependent on the underlying matrix data structure, and certain matrix types. You MUST call [MatRestoreArray\(\)](#) when you no longer need to access the array.

Synopsis

```
#include "petscmat.h"
PetscErrorCode MatGetArray(Mat mat, PetscScalar *v[])
```

Not Collective

Input Parameter

mat - the matrix

Output Parameter

v - the location of the values

Fortran Note

This routine is used differently from Fortran, e.g.,

```
Mat mat
PetscScalar mat_array(1)
PetscOffset i_mat
PetscErrorCode ierr
call MatGetArray(mat, mat_array, i_mat, ierr)
```

So, we will and it has to be also called with MatRestoreArray. So, we will do it whenever we will require it ok. So, do not worry about that ok. So, let us move on to solving this particular set of equations. So, what do we have? We want to solve we want to solve this and see here.

(Refer Slide Time: 07:10)

The image shows a code editor window on the left and a handwritten equation on the right. The code editor displays the following C code:

```
1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b;
7     int i;
8     int j[4] = {0, 1, 2, 3};
9     double ab[4] = {7.0, 1.0, 2.0, 4.0};
10    double aA[4][4] = {{1, 0, 4, 2},
11                      {2, 6, 1, 5},
12                      {0, 1, -1, -2},
13                      {4, 3, -2, 1}};
14
15    double *abb; // This will hold the values of b
16    FILE *fid;
17    PetscInitialize(&argc, &argv, NULL, "Create vector AND
18
19    VecCreate(PETSC_COMM_WORLD, &b);
20    VecSetSizes(b, PETSC_DECIDE, 4);
21    VecSetFromOptions(b);
22    VecSetValues(b, 4, j, ab, INSERT_VALUES);
23    VecAssemblyBegin(b); VecAssemblyEnd(b);
24
```

The handwritten equation on the right is:

$$\begin{bmatrix} 1 & 0 & 4 & 2 \\ 2 & 6 & 1 & 5 \\ 0 & 1 & -1 & -2 \\ 4 & 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 2 \\ 4 \end{bmatrix}$$

So, let us see how we can solve. So, it is not a very formidable set of algebraic equations it is rather easy to solve it by hand as well, but it is a bit time consuming unless you are a mathematical sound, but anyway. So, we will use.

(Refer Slide Time: 08:02)

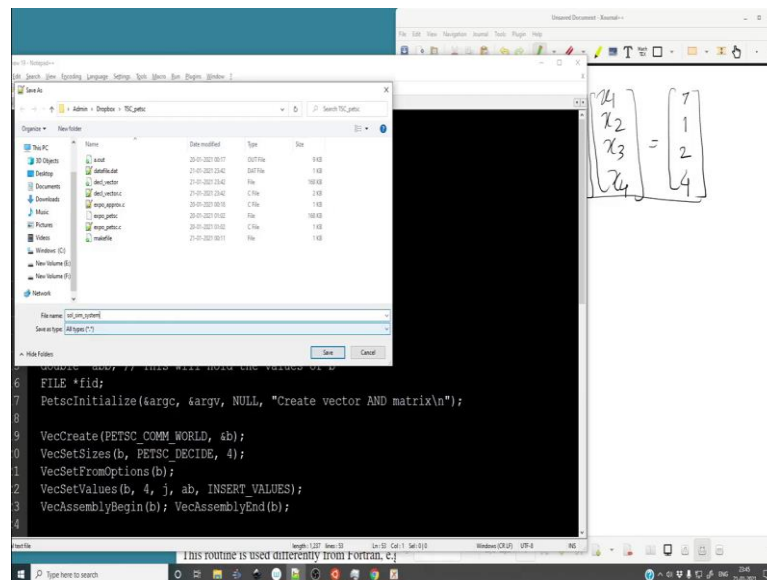
The image shows a code editor window on the left with the same code as the previous slide, but with several lines highlighted in blue. The highlighted lines are:

```
1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b;
7     int i;
8     int j[4] = {0, 1, 2, 3};
9     double ab[4] = {7.0, 1.0, 2.0, 4.0};
10    double aA[4][4] = {{1, 0, 4, 2},
11                      {2, 6, 1, 5},
12                      {0, 1, -1, -2},
13                      {4, 3, -2, 1}};
14
15    double *abb; // This will hold the values of b
16    FILE *fid;
17    PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix");
18
19    VecCreate(PETSC_COMM_WORLD, &b);
20    VecSetSizes(b, PETSC_DECIDE, 4);
21    VecSetFromOptions(b);
22    VecSetValues(b, 4, j, ab, INSERT_VALUES);
23    VecAssemblyBegin(b); VecAssemblyEnd(b);
24
```

The handwritten equation on the right is the same as in the previous slide:

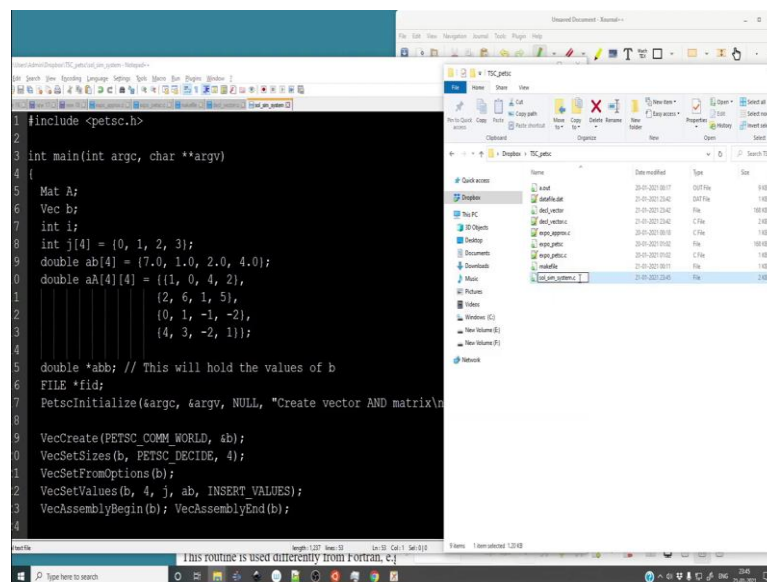
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 2 \\ 4 \end{bmatrix}$$

(Refer Slide Time: 08:07)



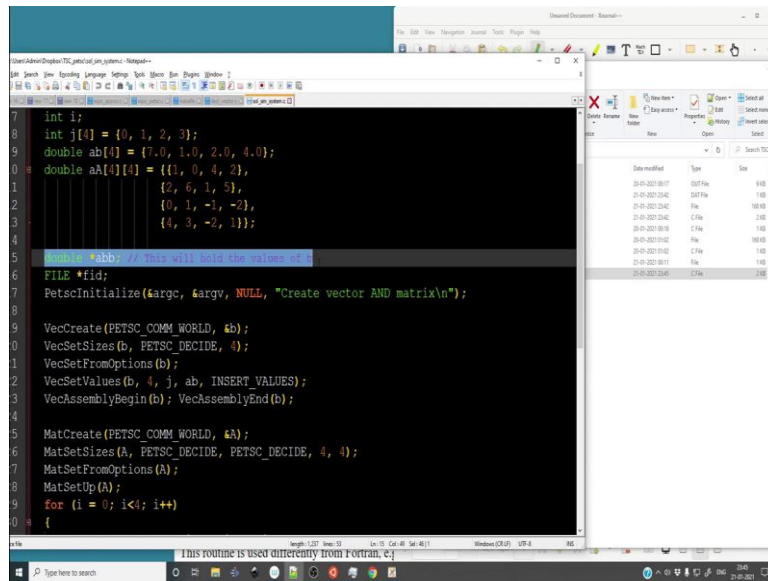
So, let us copy this code let us not. So, let me save it we will write it as solve simple sol sim system.

(Refer Slide Time: 08:25)



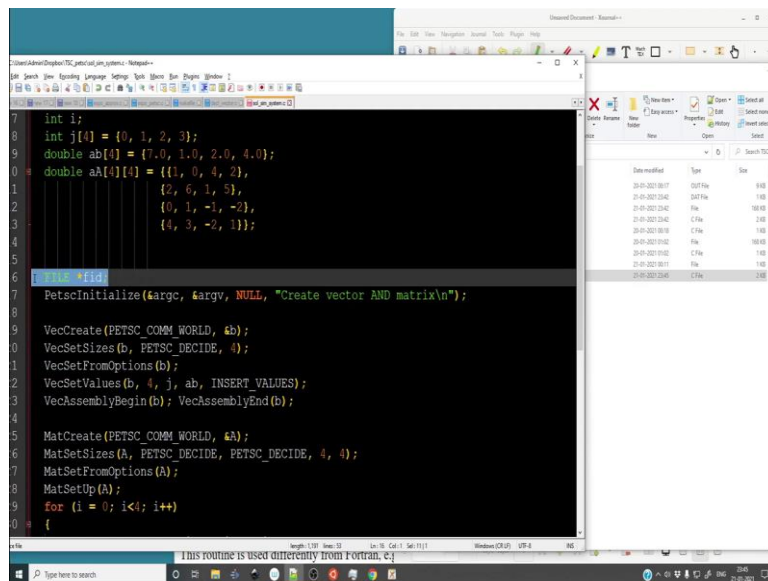
I forgot to put the extension let me just go to the folder and rename it dot c alright yes sorry alright.

(Refer Slide Time: 08:41)



```
7 int i;
8 int j[4] = {0, 1, 2, 3};
9 double ab[4] = {7.0, 1.0, 2.0, 4.0};
10 double aA[4][4] = {{1, 0, 4, 2},
11                   {2, 6, 1, 5},
12                   {0, 1, -1, -2},
13                   {4, 3, -2, 1}};
14
15 // Create file id
16 FILE *fid;
17 PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
18
19 VecCreate(PETSC_COMM_WORLD, &b);
20 VecSetSizes(b, PETSC_DECIDE, 4);
21 VecSetFromOptions(b);
22 VecSetValues(b, 4, j, ab, INSERT_VALUES);
23 VecAssemblyBegin(b); VecAssemblyEnd(b);
24
25 MatCreate(PETSC_COMM_WORLD, &A);
26 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
27 MatSetFromOptions(A);
28 MatSetUp(A);
29 for (i = 0; i < 4; i++)
30 {
```

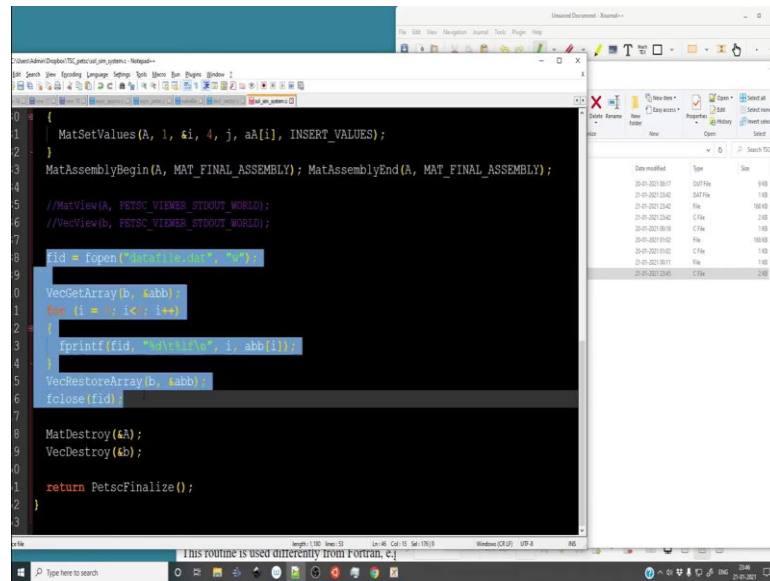
(Refer Slide Time: 08:45)



```
7 int i;
8 int j[4] = {0, 1, 2, 3};
9 double ab[4] = {7.0, 1.0, 2.0, 4.0};
10 double aA[4][4] = {{1, 0, 4, 2},
11                   {2, 6, 1, 5},
12                   {0, 1, -1, -2},
13                   {4, 3, -2, 1}};
14
15 // Create file id
16 FILE *fid;
17 PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
18
19 VecCreate(PETSC_COMM_WORLD, &b);
20 VecSetSizes(b, PETSC_DECIDE, 4);
21 VecSetFromOptions(b);
22 VecSetValues(b, 4, j, ab, INSERT_VALUES);
23 VecAssemblyBegin(b); VecAssemblyEnd(b);
24
25 MatCreate(PETSC_COMM_WORLD, &A);
26 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
27 MatSetFromOptions(A);
28 MatSetUp(A);
29 for (i = 0; i < 4; i++)
30 {
```

So, let us remove this abb business you do not need it for now let us remove the file id.

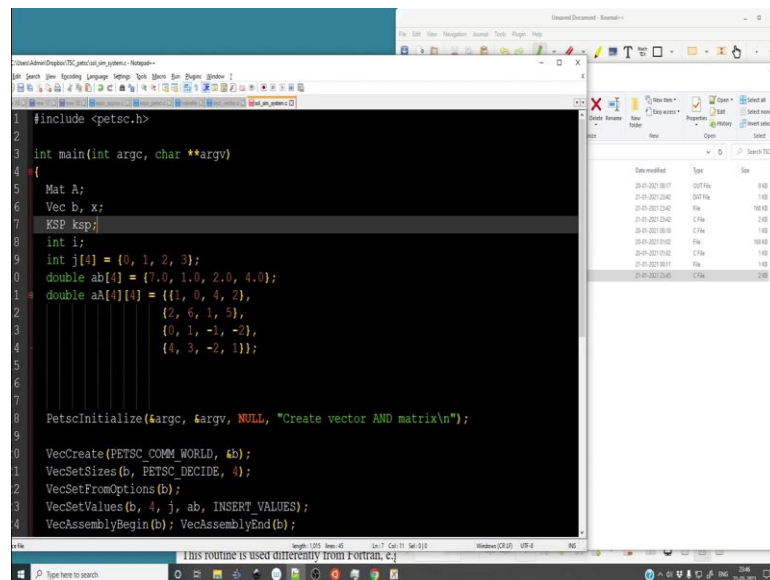
(Refer Slide Time: 08:51)



```
1 {
2     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
3 }
4 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
5
6 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
7 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
8
9 fid = fopen("matrix.txt", "w");
10 VecGetArray(b, &abb);
11 for (i = 0; i < 4; i++)
12 {
13     fprintf(fid, "%d\t", i, abb[i]);
14 }
15 VecRestoreArray(b, &abb);
16 fclose(fid);
17
18 MatDestroy(&A);
19 VecDestroy(&b);
20
21 return PetscFinalize();
22 }
```

Let us remove this bit of program we do not need it I mean if at all you need it, you can always rewrite that bit of code.

(Refer Slide Time: 08:57)

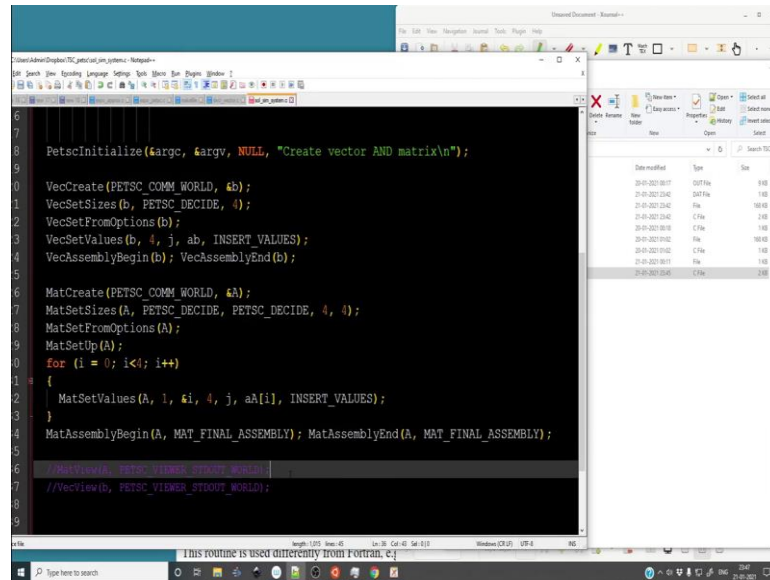


```
1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b, x;
7     KSP ksp;
8     int i;
9     int j[4] = {0, 1, 2, 3};
10    double ab[4] = {7.0, 1.0, 2.0, 4.0};
11    double aA[4][4] = {{1, 0, 4, 2},
12                      {2, 6, 1, 5},
13                      {0, 1, -1, -2},
14                      {4, 3, -2, 1}};
15
16    PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
17
18    VecCreate(PETSC_COMM_WORLD, &b);
19    VecSetSizes(b, PETSC_DECIDE, 4);
20    VecSetFromOptions(b);
21    VecSetValues(b, 4, j, ab, INSERT_VALUES);
22    VecAssemblyBegin(b); VecAssemblyEnd(b);
23 }
```

So, now we want to have another vector x which will store the solution alright and we build the array we build the vector. So, now we need to have an object which will help us actually solve the thing. So, in PETSc the solvers are a part of the object KSP ok.

So, we will declare it as KSP this is the data type and we will call it small ksp. So, ksp is the solver object solver object will operate on the vector the matrix A and the vector b to yield the solution x that is how it works.

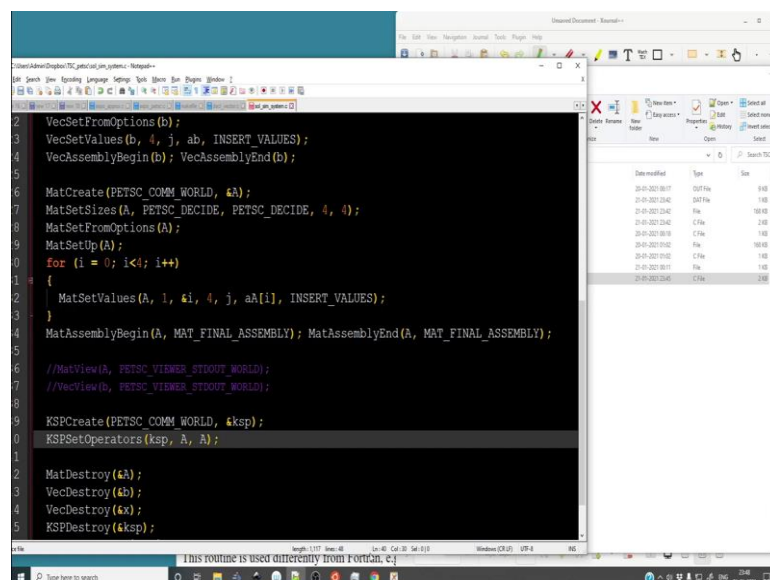
(Refer Slide Time: 09:46)



```
6
7
8 PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
9
10 VecCreate(PETSC_COMM_WORLD, &b);
11 VecSetSizes(b, PETSC_DECIDE, 4);
12 VecSetFromOptions(b);
13 VecSetValues(b, 4, j, ab, INSERT_VALUES);
14 VecAssemblyBegin(b); VecAssemblyEnd(b);
15
16 MatCreate(PETSC_COMM_WORLD, &A);
17 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
18 MatSetFromOptions(A);
19 MatSetUp(A);
20 for (i = 0; i < 4; i++)
21 {
22     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
23 }
24 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
25
26 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
27 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
28
29
```

So, similar to the matrix or the vector creation. So, this is the vector creation this is the matrix creation there has to be also a solver creation.

(Refer Slide Time: 09:59)



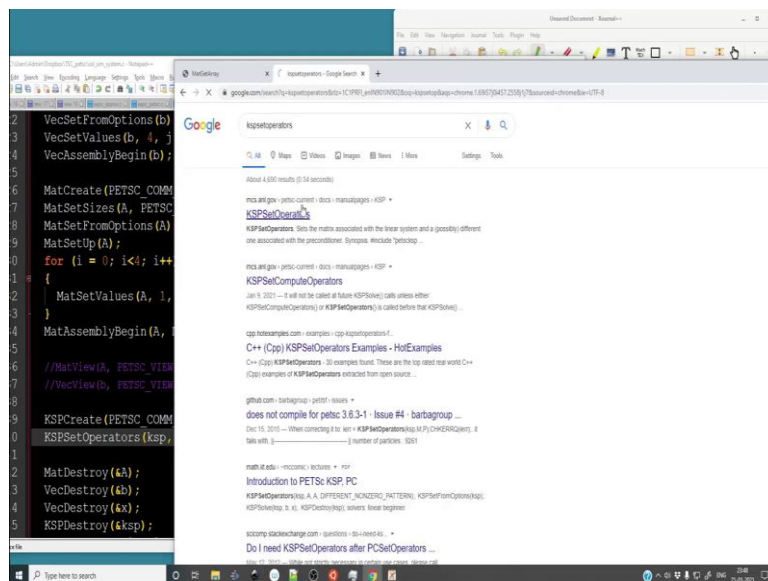
```
24 VecSetFromOptions(b);
25 VecSetValues(b, 4, j, ab, INSERT_VALUES);
26 VecAssemblyBegin(b); VecAssemblyEnd(b);
27
28 MatCreate(PETSC_COMM_WORLD, &A);
29 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
30 MatSetFromOptions(A);
31 MatSetUp(A);
32 for (i = 0; i < 4; i++)
33 {
34     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
35 }
36 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
37
38 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
39 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
40
41 KSPCreate(PETSC_COMM_WORLD, &ksp);
42 KSPSetOperators(ksp, A, A);
43
44 MatDestroy(&b);
45 VecDestroy(&b);
46 VecDestroy(&k);
47 KSPDestroy(&ksp);
48
```

So, steps are quite similar. So, it contains KSPCreate and finally. So, there is no KSPAssembly but ultimately you have to destroy. So, it there will be a KSPDestroy and

we will pass the address of ksp ok and eventually we have to also destroy x. So, these are some things you need to always do.

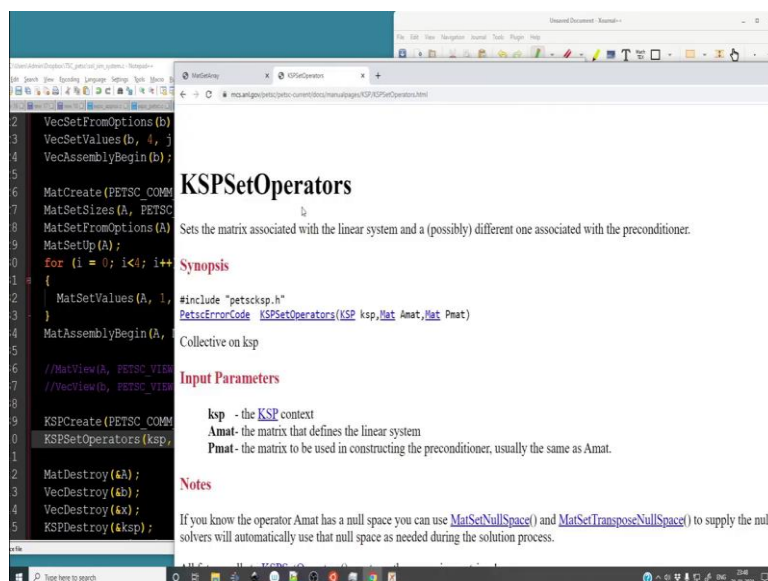
So, once we create KSP again it has to be done on com world we have to pass the address of ksp that is common. Then KSPSetOperators. So, it will have the ksp object and A, A.

(Refer Slide Time: 11:03)



So, KSPSetOperators actually sets the preconditioners let me just show you.

(Refer Slide Time: 11:09)



So, it contains the matrix which defines the linear system and the matrix to be used in the preconditioner usually it is the same as Amat that is why we have passed A, A, if it is something else you are always free to give this a different name, but in this case we are going to pass the same matrix to create the default preconditioner. So, we have set the operators.

(Refer Slide Time: 11:43)

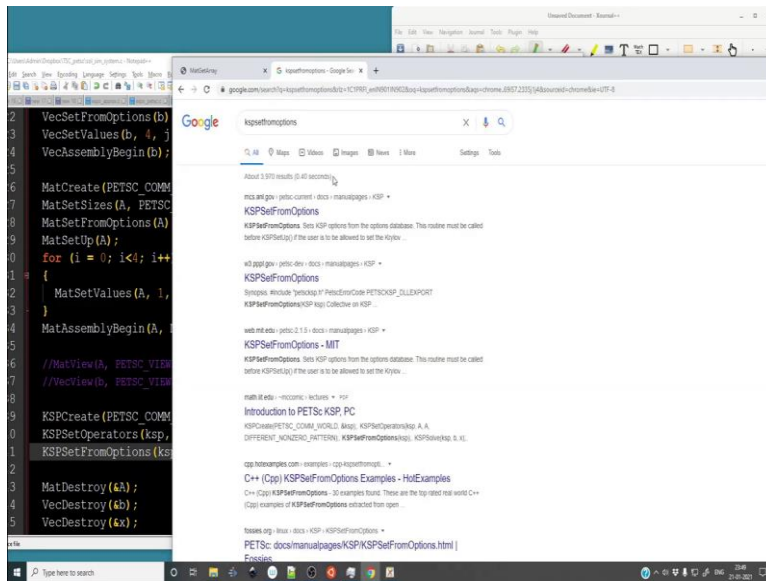
```

2 VecSetFromOptions(b);
3 VecSetValues(b, 4, j, ab, INSERT_VALUES);
4 VecAssemblyBegin(b); VecAssemblyEnd(b);
5
6 MatCreate(PETSC_COMM_WORLD, &A);
7 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
8 MatSetFromOptions(A);
9 MatSetOp(A);
10 for (i = 0; i < 4; i++)
11 {
12     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
13 }
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15
16 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
17 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
18
19 KSPCreate(PETSC_COMM_WORLD, &ksp);
20 KSPSetOperators(ksp, A, A);
21 KSPSetFromOptions(ksp);
22
23 MatDestroy(&A);
24 VecDestroy(&b);
25 VecDestroy(&x);

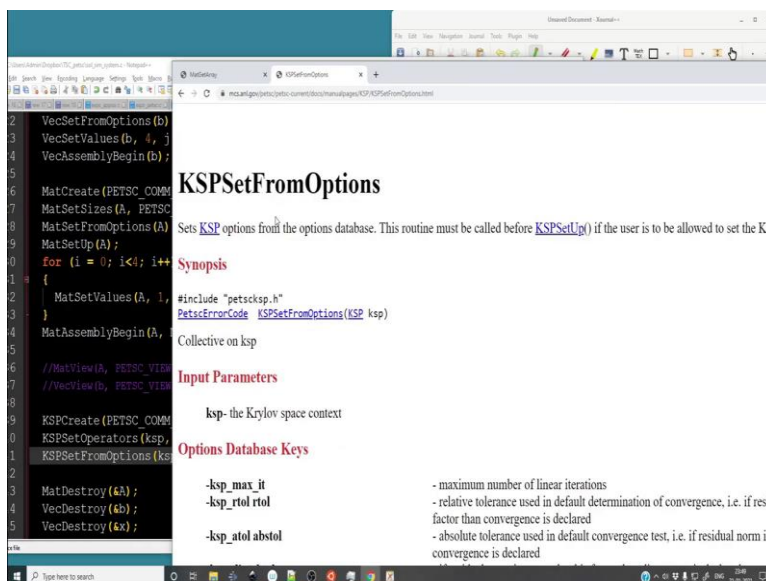
```

Then we will do KSPSetFromOptions and this will simply act on ksp and set from options is usually going to use the flags that you supply from the command line. So, various kinds of preconditioners, solvers whichever you can you want to choose you can always pass it from the command line to the code and during this particular step set from options it will set the particular options that you set.

(Refer Slide Time: 12:18)

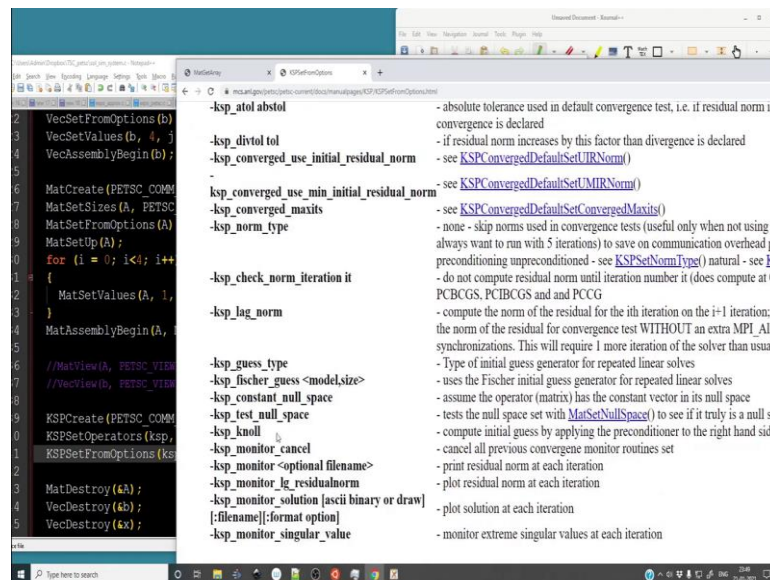


(Refer Slide Time: 12:22)



So, `kspsetfromoptions`. So, the point is the help files you can have an offline version of it, but I am keeping it and I am accessing the online version of the documentation. So, the function reference from `KSPSetFromOptions` is simply passing `ksp`.

(Refer Slide Time: 12:35)



```
2 VecSetFromOptions(b);
3 VecSetValues(b, 4, j);
4 VecAssemblyBegin(b);
5
6 MatCreate(PETSC_COMM_WORLD, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM);
7 MatSetSizes(A, PETSC_COMM_WORLD, PETSC_COMM_WORLD, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM);
8 MatSetFromOptions(A);
9 MatSetUp(A);
10 for (i = 0; i < 4; i++)
11 {
12     MatSetValues(A, 1, j, &A[i][i]);
13 }
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
15
16 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
17 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
18
19 KSPCreate(PETSC_COMM_WORLD, KSP_GMRES, KSP_GMRES);
20 KSPSetOperators(ksp, A, A);
21 KSPSetFromOptions(ksp);
22
23 MatDestroy(&A);
24 VecDestroy(&b);
25 VecDestroy(&x);
```

-ksp_atol **abstol** - absolute tolerance used in default convergence test, i.e. if residual norm is convergence is declared

-ksp_divtol - if residual norm increases by this factor than divergence is declared

-ksp_converged_use_initial_residual_norm - see [KSPConvergedDefaultSetUMIRNorm\(\)](#)

-ksp_converged_use_min_initial_residual_norm - see [KSPConvergedDefaultSetUMIRNorm\(\)](#)

-ksp_converged_maxits - see [KSPConvergedDefaultSetConvergedMaxits\(\)](#)

-ksp_norm_type - none - skip norms used in convergence tests (useful only when not using preconditioning preconditioned - see [KSPSetNormType\(\)](#)) natural - see [KSPConvergedDefaultSetUMIRNorm\(\)](#) - do not compute residual norm until iteration number it (does compute at 0th PCBCGS, PCIBCGS and PCCG)

-ksp_check_norm_iteration it - compute the norm of the residual for the ith iteration on the i+1 iteration: the norm of the residual for convergence test WITHOUT an extra MPI_Allreduce synchronizations. This will require 1 more iteration of the solver than usual.

-ksp_lag_norm - Type of initial guess generator for repeated linear solves

-ksp_guess_type - uses the Fischer initial guess generator for repeated linear solves

-ksp_fischer_guess <modelsize> - assume the operator (matrix) has the constant vector in its null space

-ksp_constant_null_space - tests the null space set with [MatSetNullSpace\(\)](#) to see if it truly is a null space

-ksp_test_null_space - compute initial guess by applying the preconditioner to the right hand side

-ksp_knoll - cancel all previous convergence monitor routines set

-ksp_monitor_cancel - print residual norm at each iteration

-ksp_monitor_optional filename> - plot residual norm at each iteration

-ksp_monitor_lg_residualnorm - plot residual norm at each iteration

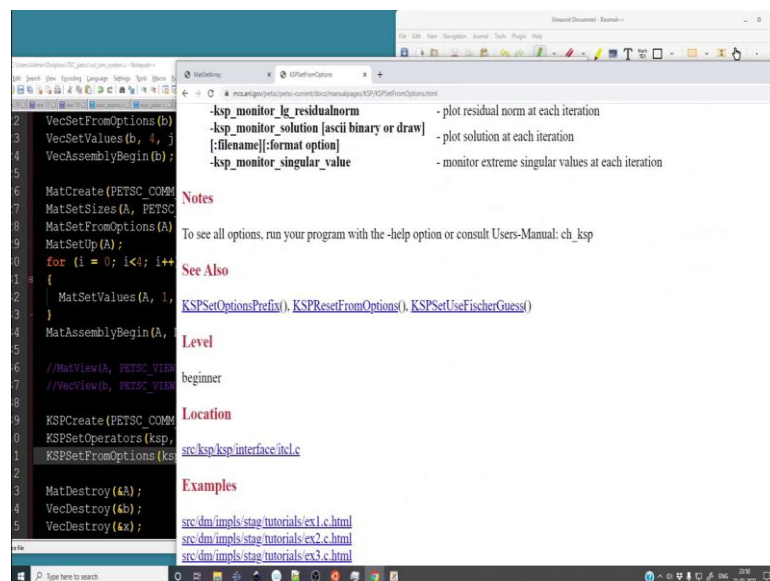
-ksp_monitor_solution [ascii binary or draw] - plot solution at each iteration

[filename]:format option

-ksp_monitor_singular_value - monitor extreme singular values at each iteration

The database options are maximum iterations, relative tolerance, absolute tolerance ok.

(Refer Slide Time: 12:45)



```
2 VecSetFromOptions(b);
3 VecSetValues(b, 4, j);
4 VecAssemblyBegin(b);
5
6 MatCreate(PETSC_COMM_WORLD, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM);
7 MatSetSizes(A, PETSC_COMM_WORLD, PETSC_COMM_WORLD, PETSC_MAT_NONSYMM, PETSC_MAT_NONSYMM);
8 MatSetFromOptions(A);
9 MatSetUp(A);
10 for (i = 0; i < 4; i++)
11 {
12     MatSetValues(A, 1, j, &A[i][i]);
13 }
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
15
16 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
17 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
18
19 KSPCreate(PETSC_COMM_WORLD, KSP_GMRES, KSP_GMRES);
20 KSPSetOperators(ksp, A, A);
21 KSPSetFromOptions(ksp);
22
23 MatDestroy(&A);
24 VecDestroy(&b);
25 VecDestroy(&x);
```

-ksp_monitor_lg_residualnorm - plot residual norm at each iteration

-ksp_monitor_solution [ascii binary or draw] - plot solution at each iteration

[filename]:format option

-ksp_monitor_singular_value - monitor extreme singular values at each iteration

Notes

To see all options, run your program with the -help option or consult Users-Manual: ch_ksp

See Also

[KSPSetOptionsPrefix\(\)](#), [KSPResetFromOptions\(\)](#), [KSPSetUseFischerGuess\(\)](#)

Level

beginner

Location

[src/ksp/kspinterface/incl.c](#)

Examples

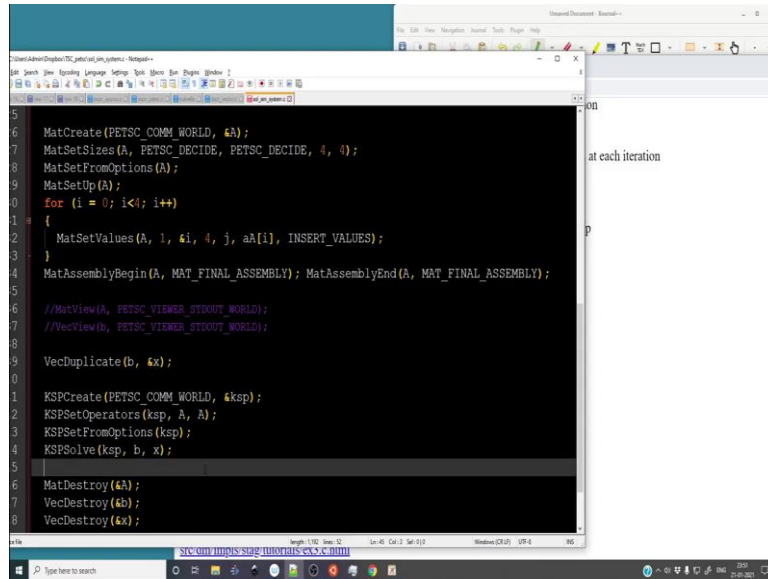
[src/dm/impls/stag/tutorials/ex1.c.html](#)

[src/dm/impls/stag/tutorials/ex2.c.html](#)

[src/dm/impls/stag/tutorials/ex3.c.html](#)

So, the guess type the monitor things like that alright. So, this is something you need to always do then what we will do is. So, before all this we need to sort of initialize what x is. So, what we will do is simply take whatever b we have and initialize x with the value of b.

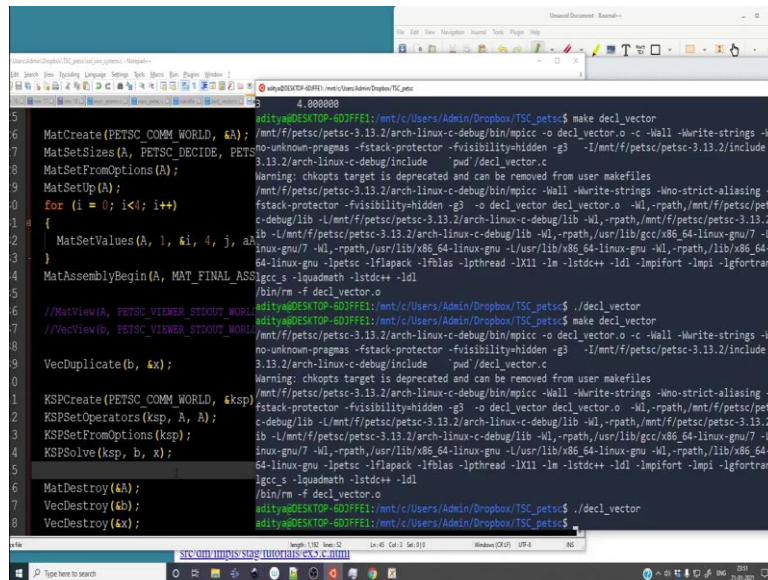
(Refer Slide Time: 13:12)



```
5
6 MatCreate(PETSC_COMM_WORLD, &A);
7 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
8 MatSetFromOptions(A);
9 MatSetUp(A);
10 for (i = 0; i < 4; i++)
11 {
12     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
13 }
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15
16 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
17 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
18
19 VecDuplicate(b, &x);
20
21 KSPCreate(PETSC_COMM_WORLD, &ksp);
22 KSPSetOperators(ksp, A, A);
23 KSPSetFromOptions(ksp);
24 KSPSolve(ksp, b, x);
25
26 MatDestroy(&A);
27 VecDestroy(&b);
28 VecDestroy(&x);
```

So, we will do VecDuplicate and we will duplicate b into x. So, we have to pass the address of x alright. So, after setting from options we will simply do KSPSolve ksp b comma x. So, ksp already has the information about the matrix A you have to pass the right hand side and the solution vector.

(Refer Slide Time: 13:50)



```
4 .000000
5 aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make decl_vector
6 /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o decl_vector.o -c -Wall -Wwrite-strings -Wno-unknown-pragmas -fstack-protector -fvisibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -3.13.2/arch-linux-c-debug/include -pud/decl_vector.c
7 Warning: chkopts target is deprecated and can be removed from user makefiles
8 /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wfstack-protector -fvisibility-hidden -g3 -o decl_vector decl_vector.o -Wl,-rpath,/mnt/f/petsc/petsc-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -lpetsc -lflapack -lfflas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lgcc_s -lquadmath -lstdc++ -ldl
9 /bin/rm -f decl_vector.o
10 aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./decl_vector
11 aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make decl_vector
12 /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o decl_vector.o -c -Wall -Wwrite-strings -Wno-unknown-pragmas -fstack-protector -fvisibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -3.13.2/arch-linux-c-debug/include -pud/decl_vector.c
13 Warning: chkopts target is deprecated and can be removed from user makefiles
14 /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wfstack-protector -fvisibility-hidden -g3 -o decl_vector decl_vector.o -Wl,-rpath,/mnt/f/petsc/petsc-debug/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -lpetsc -lflapack -lfflas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lgcc_s -lquadmath -lstdc++ -ldl
15 /bin/rm -f decl_vector.o
16 aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./decl_vector
17 aditya@DESKTOP-6DJFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, before anything let us quickly see whether we have a good compilation excellent. So, no errors fine actually this is a new file. So, we need to modify our make file.

(Refer Slide Time: 14:01)

```
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 decl_vector: decl_vector.o chkopts
13   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
14   $(RM) decl_vector.o
```

So, let us create a target for this particular for a solution system.

(Refer Slide Time: 14:09)

```
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14   $(RM) sol_sim_system.o
```

The 'Replace' dialog box shows the following configuration:

- Find: decl_vector
- Replace with: sol_sim_system
- Replace in: decl_vector.o
- Replace all:
- Replace in all Control:

So, we have to make a new target replace, replace, replace that is it.

(Refer Slide Time: 14:20)

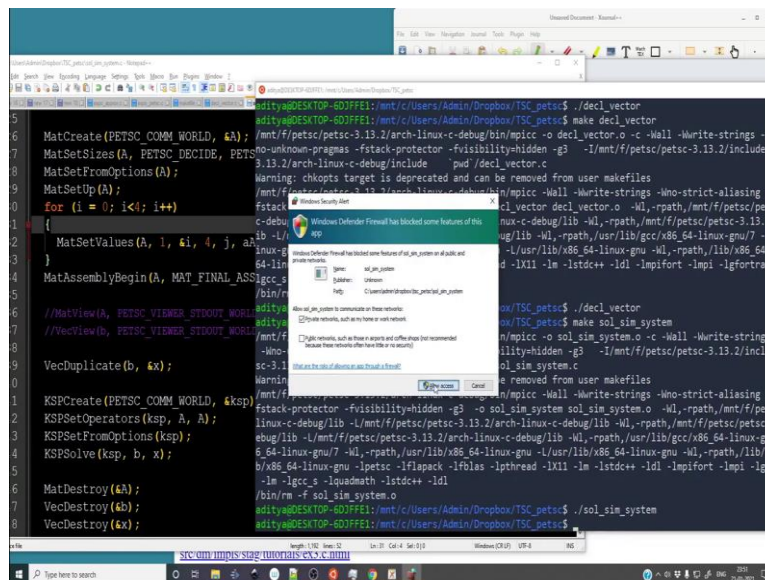
```
1 include ${PETSC_DIR}/lib/petsc/conf/variables
2 include ${PETSC_DIR}/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5 -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6 $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9 -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10 $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13 -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14 $(RM) sol_sim_system.o
```

So, we need to now compile this particular target not the previous target alright.

(Refer Slide Time: 14:28)

```
5 MatCreate(PETSC_COMM_WORLD, &A);
6 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, &M, &N);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i < N; i++)
10 {
11 MatSetValues(A, 1, &A, &A, &A, MAT_FINAL_ASSEMBLY);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 /bin/rm -f decl_vector.o
16 /bin/rm -f sol_sim_system.o
17 /bin/rm -f sol_sim_system
```

(Refer Slide Time: 14:34)

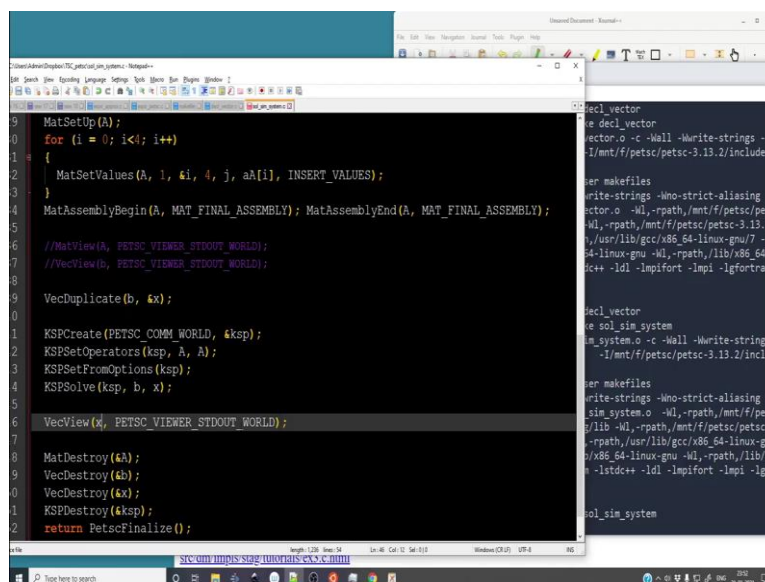


```

5 MatCreate(PETSC_COMM_WORLD, &A);
6 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, &A);
7 MatSetFromOptions(A);
8 MatSetUp(A);
9 for (i = 0; i < 4; i++)
10 {
11     MatSetValues(A, 1, &i, 4, j, A, INSERT_VALUES);
12 }
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17 VecDuplicate(b, &x);
18 KSPCreate(PETSC_COMM_WORLD, &ksp);
19 KSPSetOperators(ksp, A, A);
20 KSPSetFromOptions(ksp);
21 KSPSolve(ksp, b, x);
22 MatDestroy(&A);
23 VecDestroy(&b);
24 VecDestroy(&x);
25 KSPDestroy(&ksp);
26 return PetscFinalize();
  
```

So, we will do make same system dot slash sol sim system you have to allow this do not have to do anything if you are using you know linux by default well everything runs fine. So, now, we can finally, output the vector x.

(Refer Slide Time: 14:55)

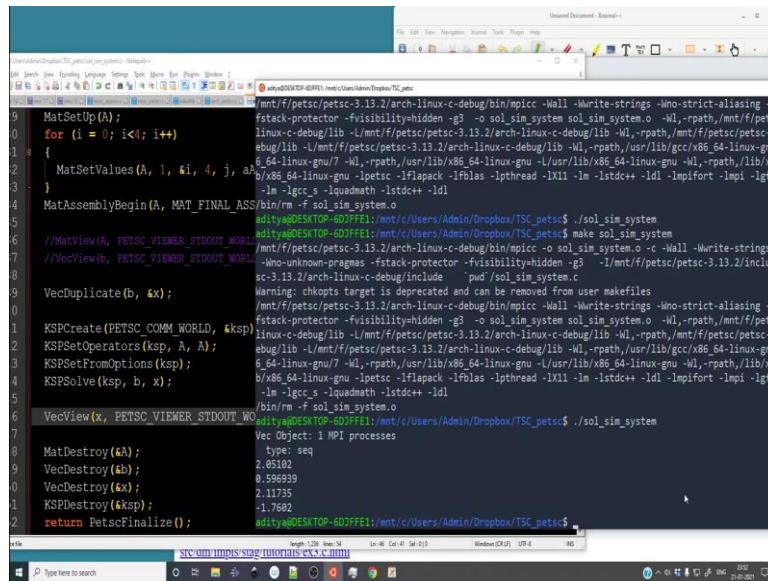


```

9 MatSetUp(A);
10 for (i = 0; i < 4; i++)
11 {
12     MatSetValues(A, 1, &i, 4, j, A, INSERT_VALUES);
13 }
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
16 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
17 VecDuplicate(b, &x);
18 KSPCreate(PETSC_COMM_WORLD, &ksp);
19 KSPSetOperators(ksp, A, A);
20 KSPSetFromOptions(ksp);
21 KSPSolve(ksp, b, x);
22 VecView(x, PETSC_VIEWER_STDOUT_WORLD);
23 MatDestroy(&A);
24 VecDestroy(&b);
25 VecDestroy(&x);
26 KSPDestroy(&ksp);
27 return PetscFinalize();
  
```

So, we can copy this back view and before destroying we can simply visualize x. So, x has to be an array of size 4.

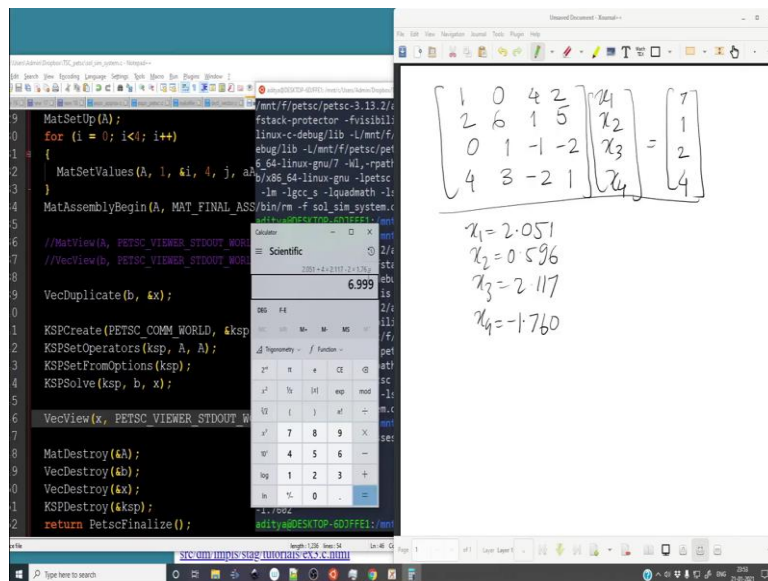
(Refer Slide Time: 15:00)



```
MatSetUp(A);
for (i = 0; i < i++;)
{
  MatSetValues(A, 1, &i, 4, j, a);
}
MatAssemblyBegin(A, MAT_FINAL_ASS/bin/rm -f sol_sim_system.o
aditya@DESKTOP-6DJFFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petac$ ./sol_sim_system
//MatView(A, PETSC_VIEWER_STDOUT_WORLD);
//VecView(b, PETSC_VIEWER_STDOUT_WORLD);
VecDuplicate(b, &x);
KSPCreate(PETSC_COMM_WORLD, &ksp);
KSPSetOperators(ksp, A, A);
KSPSetFromOptions(ksp);
KSPSolve(ksp, b, x);
VecView(x, PETSC_VIEWER_STDOUT_WORLD);
MatDestroy(&A);
VecDestroy(&b);
VecDestroy(&x);
KSPDestroy(&ksp);
return PetscFinalize();
aditya@DESKTOP-6DJFFFE1:/mnt/c/Users/Admin/Dropbox/TSC_petac$
```

So, let me recompile. So, this is the solution that we obtained alright.

(Refer Slide Time: 15:14)



```
MatSetUp(A);
for (i = 0; i < i++;)
{
  MatSetValues(A, 1, &i, 4, j, a);
}
MatAssemblyBegin(A, MAT_FINAL_ASS/bin/rm -f sol_sim_system.c
aditya@DESKTOP-6DJFFFE1:/mnt
Calculator
Scientific
6.999
x1 = 2.051
x2 = 0.596
x3 = 2.117
x4 = -1.760
```

$$\begin{bmatrix} 1 & 0 & 4 & 2 \\ 2 & 6 & 1 & 5 \\ 0 & 1 & -1 & -2 \\ 4 & 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 2 \\ 4 \end{bmatrix}$$

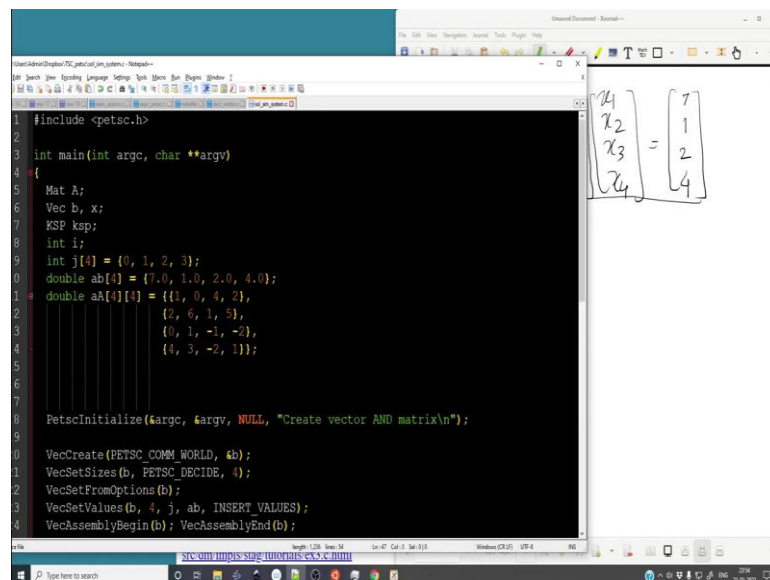
So, x1 is 2.051, x2 is 0.596, x3 is 2.117, x4 is -1.760 ok I am just taking three decimals let us quickly see whether it checks out I am shamelessly opening a calculator, but anyway let us check for the first equation $2.051 + 4 * x_3$ which is $2.117 - 2 * 1.760$, 6.999 it is as good as 7.

So, I am getting it at three decimal places, but you get the point. So, far we have made a KSP object. So, KSP stands for crew of subspace. So, if you go into the area of solving

large matrices you necessarily have to go into the theories of various kinds of solvers and invariably you will end up with solving with the help of Krylov subspace. So, Krylov subspace contains a lot of methods inside, but they are all part of what they call as Krylov subspace.

So, we are not going to discuss the theory of all these, they are left for you for a more specialized course on linear algebra perhaps, but once you start doing the course on linear algebra what you can do is make a detour of all those problems that you do you can try to numerically experiment with the help of PETSc; PETSc allows you that massively scalable set of functions data structures libraries if you will and you can do a lot of things you can experiment a lot of things.

(Refer Slide Time: 17:21)



```
1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b, x;
7     KSP ksp;
8     int i;
9     int j[4] = {0, 1, 2, 3};
10    double ab[4] = {7.0, 1.0, 2.0, 4.0};
11    double aA[4][4] = {{1, 0, 4, 2},
12                      {2, 6, 1, 5},
13                      {0, 1, -1, -2},
14                      {4, 3, -2, 1}};
15
16    PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");
17
18    VecCreate(PETSC_COMM_WORLD, &b);
19    VecSetSizes(b, PETSC_DECIDE, 4);
20    VecSetFromOptions(b);
21    VecSetValues(b, 4, j, ab, INSERT_VALUES);
22    VecAssemblyBegin(b); VecAssemblyEnd(b);
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 2 \\ 4 \end{bmatrix}$$

So, this completes the creation of vector matrix the display or the printing of a vector to a file and solving a system using a ksp object. So, in the next example what we are going to do in. In fact, before going to that what we can do.

(Refer Slide Time: 17:49)

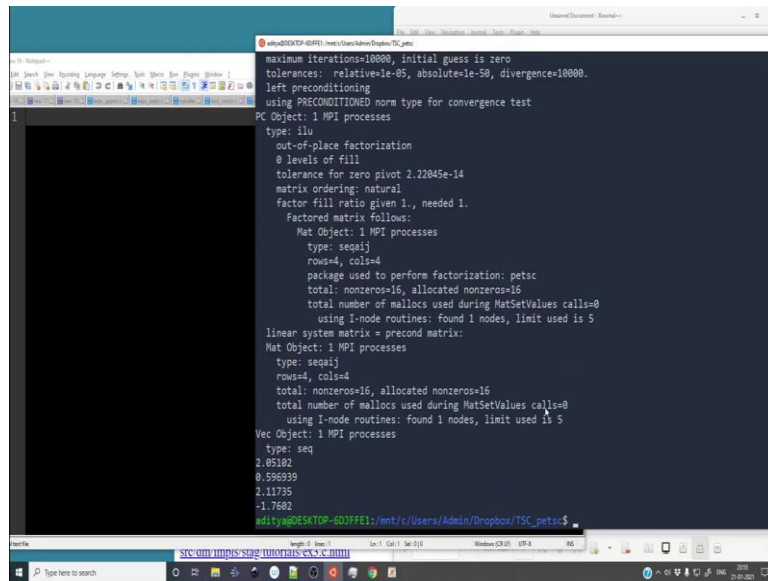
```
Warning: chkopts target is deprecated and can be removed from user makefiles
/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/epicc -Wall -Wwrite-strings -Wno-strict-aliasing -W
fstack-protector -fvisibility-hidden -g3 -o sol_sim_system sol_sim_system.o -Wl,-rpath,/mnt/ff/pets
linux-c-debug/lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3
ebug/lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu
6.4-linux-gnu/7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x8
6_64-linux-gnu -lpetsc -lflapack -lfflas -lpthread -lm11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfso
-lm -lgcc_s -liquidmath -lstdc++ -ldl
/bin/rm -f sol_sim_system.o
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system
Vec Object: 1 MPI processes
type: seq
2.85182
0.596939
2.11735
-1.7682
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -vec_view
Vec Object: 1 MPI processes
type: seq
7.
1.
2.
4.
Vec Object: 1 MPI processes
type: seq
2.85182
0.596939
2.11735
-1.7682
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, what we had was we had seen vec view something like this. So, it showed the different vectors that we had and there was also a mat view something like this.

(Refer Slide Time: 18:03)

```
0.596939
2.11735
-1.7682
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -vec_view
Vec Object: 1 MPI processes
type: seq
7.
1.
2.
4.
Vec Object: 1 MPI processes
type: seq
2.85182
0.596939
2.11735
-1.7682
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -mat_view
Mat Object: 1 MPI processes
type: seqobj
row 0: (0, 1) (1, 0.) (2, 4.) (3, 2.)
row 1: (0, 2.) (1, 6.) (2, 1.) (3, 5.)
row 2: (0, 0.) (1, 1.) (2, -1.) (3, -2.)
row 3: (0, 4.) (1, 3.) (2, -2.) (3, 1.)
Vec Object: 1 MPI processes
type: seq
2.85182
0.596939
2.11735
-1.7682
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

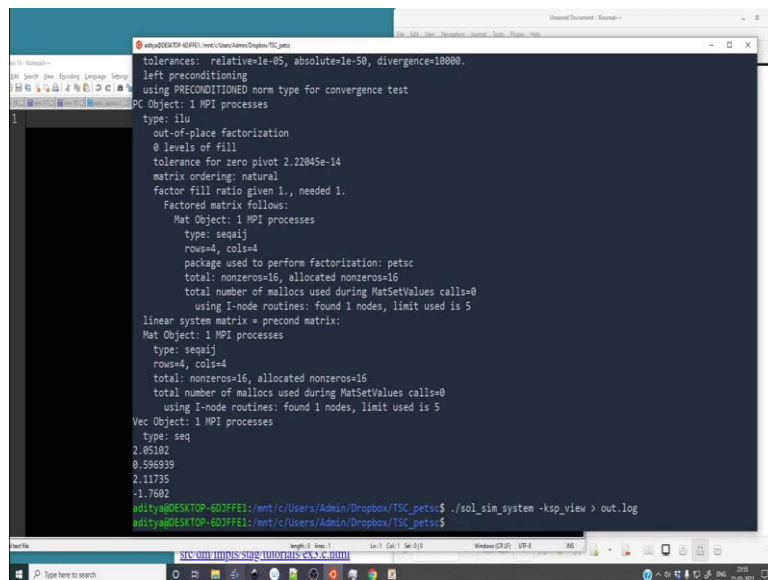
(Refer Slide Time: 18:12)



```
Maximum iterations=10000, initial guess is zero
tolerances: relative=1e-05, absolute=1e-50, divergence=10000.
left preconditioning
using PRECONDITIONED norm type for convergence test
PC Object: 1 MPI processes
type: ilu
out-of-place factorization
0 levels of fill
tolerance for zero pivot 2.22045e-14
matrix ordering: natural
factor fill ratio given 1., needed 1.
Factored matrix follows:
Mat Object: 1 MPI processes
type: seqaij
rows=4, cols=4
package used to perform factorization: petsc
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
linear system matrix = precond matrix:
Mat Object: 1 MPI processes
type: seqaij
rows=4, cols=4
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

But there is also a ksp view that shows us the information about the method used. So, what do we have? Unable to scroll up.

(Refer Slide Time: 18:36)



```
tolerances: relative=1e-05, absolute=1e-50, divergence=10000.
left preconditioning
using PRECONDITIONED norm type for convergence test
PC Object: 1 MPI processes
type: ilu
out-of-place factorization
0 levels of fill
tolerance for zero pivot 2.22045e-14
matrix ordering: natural
factor fill ratio given 1., needed 1.
Factored matrix follows:
Mat Object: 1 MPI processes
type: seqaij
rows=4, cols=4
package used to perform factorization: petsc
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
linear system matrix = precond matrix:
Mat Object: 1 MPI processes
type: seqaij
rows=4, cols=4
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -ksp_view > out.log
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

What we can do is; pipe it to a file we can look at the log file.

(Refer Slide Time: 18:44)

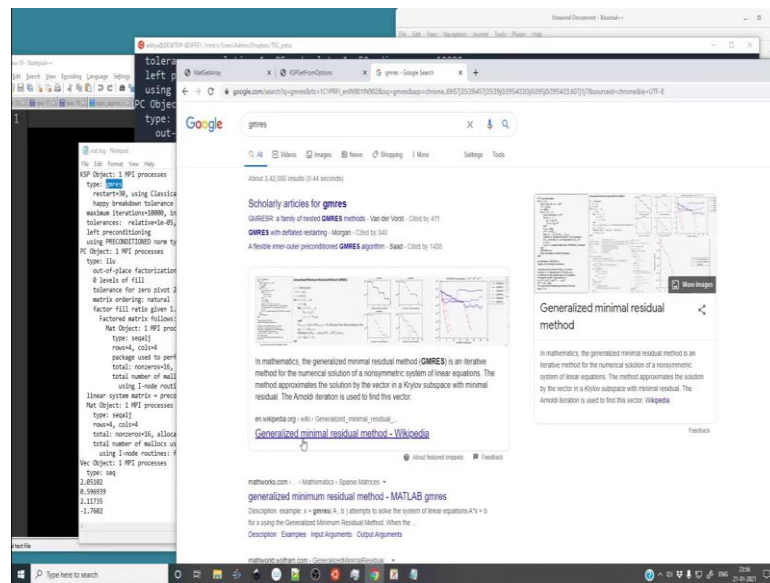
```
tolerances: relative=1e-05, absolute=1e-50, divergence=10000
left preconditioning
using PRECONDITIONED norm type for convergence test
KSP Object: 1 MPI processes
type: ilu
out-of-place factorization
0 levels of fill
tolerance for zero pivot 2.22045e-14
matrix ordering: natural
factor fill ratio given 1., needed 1.
Factored matrix follows:
Mat Object: 1 MPI processes
type: seqajj
rows=4, cols=4
package used to perform factorization: petsc
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues
using I-node routines: found 1 nodes, limit used is 5
linear system matrix = preconditioned matrix:
Mat Object: 1 MPI processes
type: seqajj
rows=4, cols=4
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=
using I-node routines: found 1 nodes, limit used is 5
Vec Object: 1 MPI processes
type: seq
2.85182
0.595939
2.11735
-1.7502
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc
```

(Refer Slide Time: 18:47)

```
tolerances: relative=1e-05, absolute=1e-50, divergence=10000
left preconditioning
using PRECONDITIONED norm type for convergence test
KSP Object: 1 MPI processes
type: gmres
out-of-place factorization
0 levels of fill
tolerance for zero pivot 2.22045e-14
matrix ordering: natural
factor fill ratio given 1., needed 1.
Factored matrix follows:
Mat Object: 1 MPI processes
type: seqajj
rows=4, cols=4
package used to perform factorization: petsc
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
linear system matrix = preconditioned matrix:
Mat Object: 1 MPI processes
type: seqajj
rows=4, cols=4
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
Vec Object: 1 MPI processes
type: seq
2.85182
0.595939
2.11735
-1.7502
aditya@DESKTOP-603FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc
```

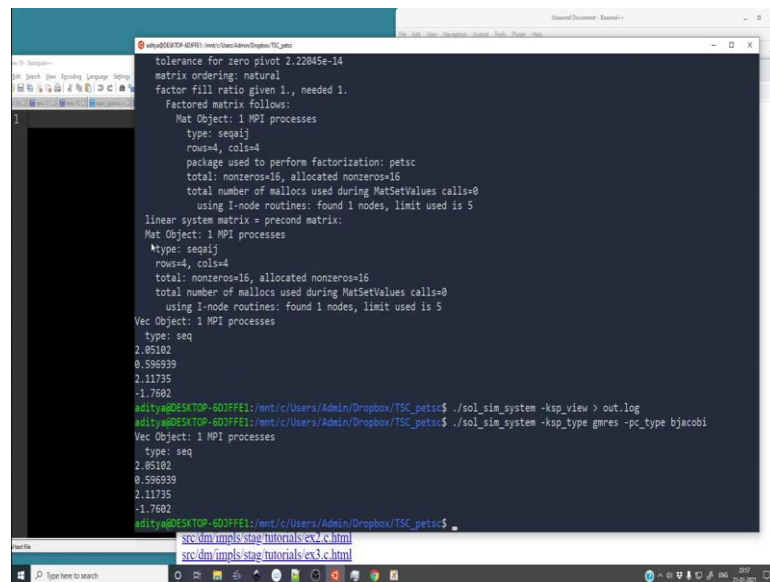
So, KSP object one MPI it is using a gmres. So, gmres stands for generalized minimum well I forget the full form gmres generalized minimum residual method ok.

(Refer Slide Time: 19:00)



So, that particular method has certain number of restarts, it has a breakdown tolerance maximum of iterations PRECONDITIONED norm for convergence test it is using a PRECONDITIONED it is using `ilu` in complete `lu` decomposition as the PRECONDITIONED and it shows all the information that you need ok.

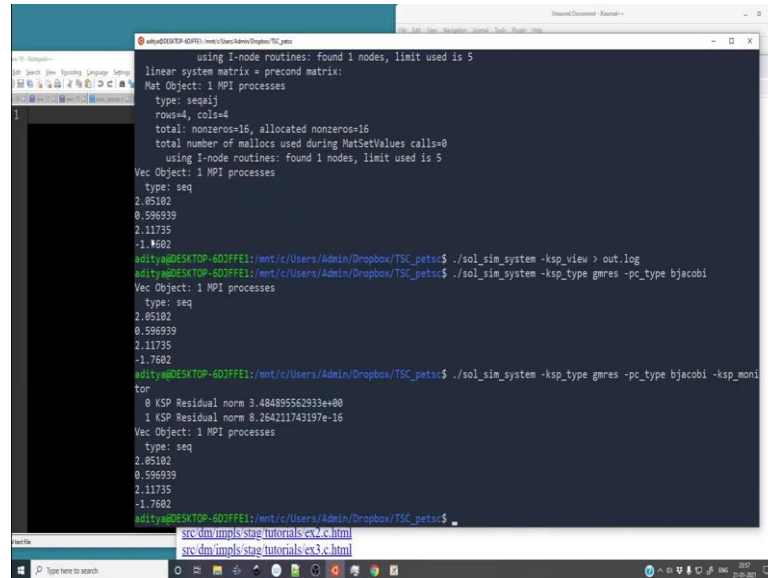
(Refer Slide Time: 19:46)



So, what we can do? In fact, is change the solver type at runtime. So, what we can we see we can change various options. In fact, we can change various options we can do - `ksp`

type gmres - pc type bjacobi block jacobi let us see and; obviously, gives the same answer, but it is uses a different pre-conditioner.

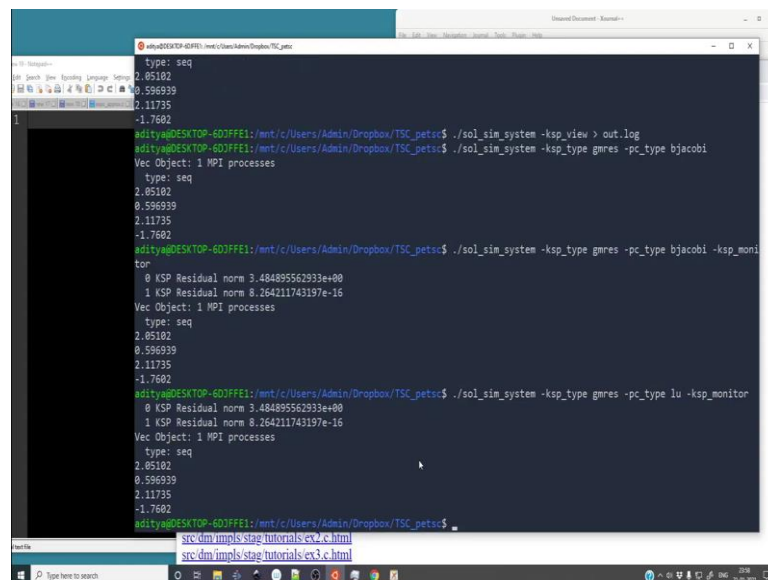
(Refer Slide Time: 20:25)



```
using I-node routines: found 1 nodes, limit used is 5
linear system matrix = precondition matrix:
Mat Object: 1 MPI processes
type: seqaij
rows=4, cols=4
total: nonzeros=16, allocated nonzeros=16
total number of mallocs used during MatSetValues calls=0
using I-node routines: found 1 nodes, limit used is 5
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_view > out.log
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_type gmres -pc_type bjacobi
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_type gmres -pc_type bjacobi -ksp_monitor
0 KSP Residual norm 3.484895562933e+00
1 KSP Residual norm 8.2642111743197e-16
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$
```

We can also do a minus ksp monitor to see how the progress is. So, immediately after many iteration we have convergence this is expected for such a small system ok. We can use a preconditioner which is not the ilu but in fact, a simple lu decomposition.

(Refer Slide Time: 20:41)



```
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_view > out.log
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_type gmres -pc_type bjacobi
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_type gmres -pc_type bjacobi -ksp_monitor
0 KSP Residual norm 3.484895562933e+00
1 KSP Residual norm 8.2642111743197e-16
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$ ./sol_sim_system -ksp_type gmres -pc_type lu -ksp_monitor
0 KSP Residual norm 3.484895562933e+00
1 KSP Residual norm 8.2642111743197e-16
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petr$
```

So, we can change various options at runtime ok.

(Refer Slide Time: 20:56)

```
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
-ksp_final_residual saws[:communicatorname]: Publishes object to SAsWs (PetscOptionsGetViewer)
-ksp_pc_side <now LEFT : formerly LEFT> KSP preconditioner side (choose one of) LEFT RIGHT SYMMETRIC (KSPSetPCSide)
KSP GRES Options
-ksp_gmres_restart <now 30 : formerly 30>: Number of Krylov search directions (KSPGMRESRestart)
-ksp_gmres_hapitol <1e-30 : 1e-30>: Tolerance for exact convergence (happy ending) (KSPGMRESSetHapitol)
-ksp_gmres_preallocate: <FALSE : FALSE> Preallocate Krylov vectors (KSPGMRESSetPreAllocateVectors)
Pick at most one of -----
-ksp_gmres_classicalgramschmidt: Classical (unmodified) Gram-Schmidt (fast) (KSPGMRESSetOrthogonalization)
-ksp_gmres_modifiedgramschmidt: Modified Gram-Schmidt (slow,more stable) (KSPGMRESSetOrthogonalization)
-ksp_gmres_cgs_refinement_type <now REFINE_NEVER : formerly REFINE_NEVER>: Type of iterative refinement for classical (unmodified) Gram-Schmidt (choose one of) REFINE_NEVER REFINE_IFNEEDED REFINE_ALWAYS (KSPGMRESSetCGSRefinementType)
-ksp_gmres_krylov_monitor: <FALSE : FALSE> Plot the Krylov directions (KSPMonitorSet)
-----
Viewer (-is_view) options:
-is_view_ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
-is_view_binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
-is_view_draw[:drawtype][:filename[:format]]]: Draws object (PetscOptionsGetViewer)
-is_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-is_view_saws[:communicatorname]: Publishes object to SAsWs (PetscOptionsGetViewer)
-----
Viewer (-mat_factor_view) options:
-mat_factor_view_ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
-mat_factor_view_binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
-mat_factor_view_draw[:drawtype][:filename[:format]]]: Draws object (PetscOptionsGetViewer)
-mat_factor_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-mat_factor_view_saws[:communicatorname]: Publishes object to SAsWs (PetscOptionsGetViewer)
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

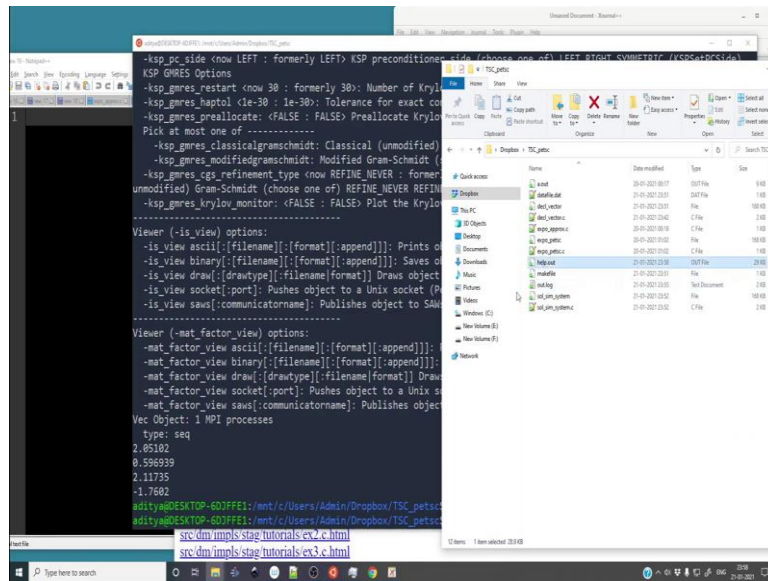
So, what are the different options available to us let us see; let us see minus help. So, it shows all the different ok.

(Refer Slide Time: 21:08)

```
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -help > help.out
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
-ksp_pc_side <now LEFT : formerly LEFT> KSP preconditioner side (choose one of) LEFT RIGHT SYMMETRIC (KSPSetPCSide)
KSP GRES Options
-ksp_gmres_restart <now 30 : formerly 30>: Number of Krylov search directions (KSPGMRESRestart)
-ksp_gmres_hapitol <1e-30 : 1e-30>: Tolerance for exact convergence (happy ending) (KSPGMRESSetHapitol)
-ksp_gmres_preallocate: <FALSE : FALSE> Preallocate Krylov vectors (KSPGMRESSetPreAllocateVectors)
Pick at most one of -----
-ksp_gmres_classicalgramschmidt: Classical (unmodified) Gram-Schmidt (fast) (KSPGMRESSetOrthogonalization)
-ksp_gmres_modifiedgramschmidt: Modified Gram-Schmidt (slow,more stable) (KSPGMRESSetOrthogonalization)
-ksp_gmres_cgs_refinement_type <now REFINE_NEVER : formerly REFINE_NEVER>: Type of iterative refinement for classical (unmodified) Gram-Schmidt (choose one of) REFINE_NEVER REFINE_IFNEEDED REFINE_ALWAYS (KSPGMRESSetCGSRefinementType)
-ksp_gmres_krylov_monitor: <FALSE : FALSE> Plot the Krylov directions (KSPMonitorSet)
-----
Viewer (-is_view) options:
-is_view_ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
-is_view_binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
-is_view_draw[:drawtype][:filename[:format]]]: Draws object (PetscOptionsGetViewer)
-is_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-is_view_saws[:communicatorname]: Publishes object to SAsWs (PetscOptionsGetViewer)
-----
Viewer (-mat_factor_view) options:
-mat_factor_view_ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
-mat_factor_view_binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
-mat_factor_view_draw[:drawtype][:filename[:format]]]: Draws object (PetscOptionsGetViewer)
-mat_factor_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-mat_factor_view_saws[:communicatorname]: Publishes object to SAsWs (PetscOptionsGetViewer)
Vec Object: 1 MPI processes
type: seq
2.05102
0.596939
2.11735
-1.7602
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

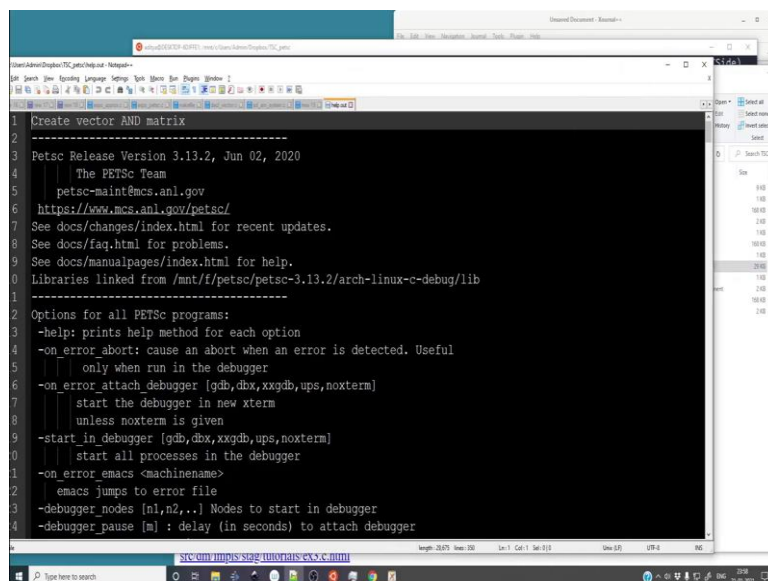
I cannot scroll minus help. In fact, let us get only. So, it has various kinds of. So, let us pipe it to a file let us pipe it to a file. So, that we can open it easily.

(Refer Slide Time: 21:27)



```
aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./ksp -help
ksp -help
-----
-ksp_pc_side <now LEFT : formerly LEFT> KSP preconditioner (choose one of LEFT RIGHT CVMGTRIV / (VOC+P+G+Z))
-ksp_gmres_restart <now 30 : formerly 30>: Number of Krylov
-ksp_gmres_hstol <1e-30 : 1e-30>: Tolerance for exact convergence
-ksp_gmres_preallocate <FALSE : FALSE> Preallocate Krylov
Pick at most one of -----
-ksp_gmres_classicalgramschmidt: Classical (unmodified)
-ksp_gmres_modifiedgramschmidt: Modified Gram-Schmidt (
-ksp_gmres_cgs_refinement_type <now REFINE_NEVER : former
unmodified> Gram-Schmidt (choose one of REFINE_NEVER REFIN
-ksp_gmres_krylov_monitor <FALSE : FALSE> Plot the Krylov
-----
Viewer (-is_view) options:
-is_view_ascii[:filename][:format][:append]]: Prints o
-is_view_binary[:filename][:format][:append]]: Saves o
-is_view_draw[:drawtype][:filename[:format]]: Draws object
-is_view_socket[:port]: Pushes object to a Unix socket (P
-is_view_saw[:communicatorname]: Publishes object to SMC
-----
Viewer (-mat_factor_view) options:
-mat_factor_view_ascii[:filename][:format][:append]]:
-mat_factor_view_binary[:filename][:format][:append]]:
-mat_factor_view_draw[:drawtype][:filename[:format]]: Draw
-mat_factor_view_socket[:port]: Pushes object to a Unix s
-mat_factor_view_saws[:communicatorname]: Publishes objec
Vec Object: 1 MPI processes
type: seq
2.65182
0.596939
2.11735
-1.7602
aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

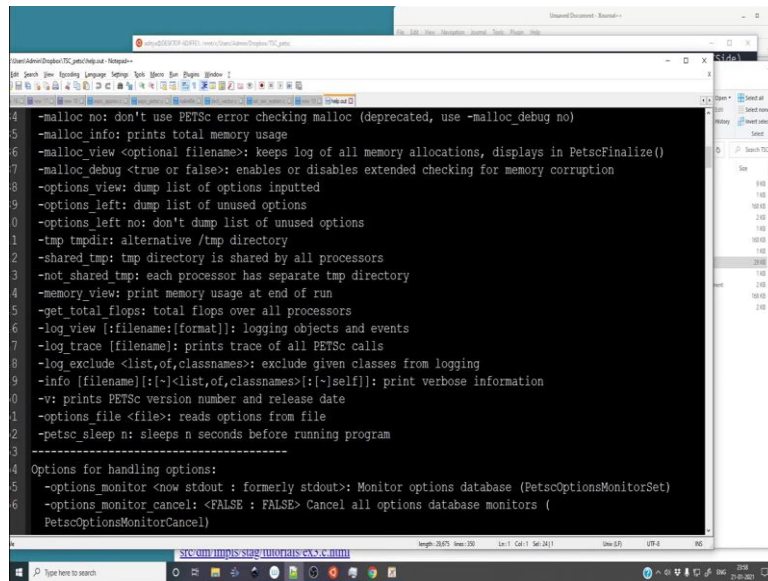
(Refer Slide Time: 21:34)



```
aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./help
Create vector AND matrix
-----
Petsc Release Version 3.13.2, Jun 02, 2020
The PETSc Team
petsc-maint@mcs.anl.gov
https://www.mcs.anl.gov/petsc/
See docs/changes/index.html for recent updates.
See docs/faq.html for problems.
See docs/manualpages/index.html for help.
Libraries linked from /mnt/£/petsc/petsc-3.13.2/arch-linux-c-debug/lib
-----
Options for all PETSc programs:
-help: prints help method for each option
-on_error_abort: cause an abort when an error is detected. Useful
only when run in the debugger
-on_error_attach_debugger [gdb,dbx,xxgdb,ups,noxterm]
start the debugger in new xterm
unless noxterm is given
-start_in_debugger [gdb,dbx,xxgdb,ups,noxterm]
start all processes in the debugger
-on_error_emacs <machinename>
emacs jumps to error file
-debugger_nodes [n1,n2,..] Nodes to start in debugger
-debugger_pause [n] : delay (in seconds) to attach debugger
-----
length: 3873, lines: 300, len: 1, Col: 1, Scl: 310, Uln: 0, UPl: 0, RCl:
aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

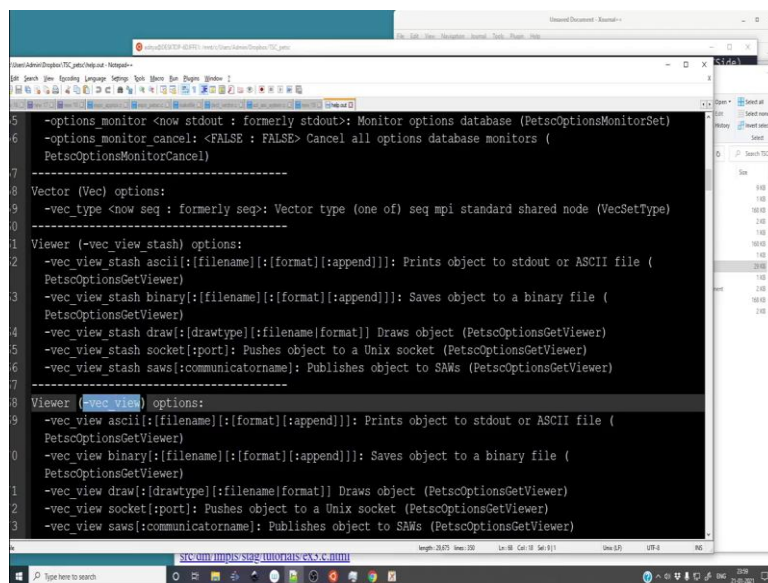
So, help dot out. So, what does help dot out contain it contains the help string that we put then it contains a bunch of options that you can pass.

(Refer Slide Time: 21:43)



```
4 -malloc no: don't use PETSc error checking malloc (deprecated, use -malloc_debug no)
5 -malloc_info: prints total memory usage
6 -malloc_view <optional filename>: keeps log of all memory allocations, displays in PetscFinalize()
7 -malloc_debug <true or false>: enables or disables extended checking for memory corruption
8 -options_view: dump list of options inputted
9 -options_left: dump list of unused options
10 -options_left no: don't dump list of unused options
11 -tmp tmpdir: alternative /tmp directory
12 -shared_tmp: tmp directory is shared by all processors
13 -not_shared_tmp: each processor has separate tmp directory
14 -memory_view: print memory usage at end of run
15 -get_total_flops: total flops over all processors
16 -log_view [:filename:[format]]: logging objects and events
17 -log_trace [filename]: prints trace of all PETSc calls
18 -log_exclude <list,of,classnames>: exclude given classes from logging
19 -info [filename][:[:<list,of,classnames>[:[:self]]]: print verbose information
20 -v: prints PETSc version number and release date
21 -options_file <file>: reads options from file
22 -petsc_sleep n: sleeps n seconds before running program
23
24 -----
25 Options for handling options:
26 -options_monitor <now stdout : formerly stdout>: Monitor options database (PetscOptionsMonitorSet)
27 -options_monitor_cancel: <FALSE : FALSE> Cancel all options database monitors (
28   PetscOptionsMonitorCancel)
29
30 -----
```

(Refer Slide Time: 21:49)



```
5 -options_monitor <now stdout : formerly stdout>: Monitor options database (PetscOptionsMonitorSet)
6 -options_monitor_cancel: <FALSE : FALSE> Cancel all options database monitors (
7   PetscOptionsMonitorCancel)
8
9 -----
10 Vector (-vec) options:
11 -vec_type <now seq : formerly seq>: Vector type (one of) seq mpi standard shared node (VecSetType)
12
13 -----
14 Viewer (-vec_view_stash) options:
15 -vec_view_stash_ascii[:[:filename[:[:format[:[:append]]]]]: Prints object to stdout or ASCII file (
16   PetscOptionsGetViewer)
17 -vec_view_stash_binary[:[:filename[:[:format[:[:append]]]]]: Saves object to a binary file (
18   PetscOptionsGetViewer)
19 -vec_view_stash_draw[:[:drawtype[:[:filename[:format]]]]: Draws object (PetscOptionsGetViewer)
20 -vec_view_stash_socket[:[:port]]: Pushes object to a Unix socket (PetscOptionsGetViewer)
21 -vec_view_stash_saws[:[:communicatorname]]: Publishes object to SAs (PetscOptionsGetViewer)
22
23 -----
24 Viewer (-vec_view) options:
25 -vec_view_ascii[:[:filename[:[:format[:[:append]]]]]: Prints object to stdout or ASCII file (
26   PetscOptionsGetViewer)
27 -vec_view_binary[:[:filename[:[:format[:[:append]]]]]: Saves object to a binary file (
28   PetscOptionsGetViewer)
29 -vec_view_draw[:[:drawtype[:[:filename[:format]]]]: Draws object (PetscOptionsGetViewer)
30 -vec_view_socket[:[:port]]: Pushes object to a Unix socket (PetscOptionsGetViewer)
31 -vec_view_saws[:[:communicatorname]]: Publishes object to SAs (PetscOptionsGetViewer)
32
33 -----
```


(Refer Slide Time: 21:55)

```
PetscOptionsGetViewer)
0  -vec_view binary[:filename][:format][:append]]: Saves object to a binary file (
  PetscOptionsGetViewer)
1  -vec_view draw[:drawtype][:filename|format]] Draws object (PetscOptionsGetViewer)
2  -vec_view socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
3  -vec_view saws[:communicatorname]: Publishes object to SAMs (PetscOptionsGetViewer)
4  -----
5  Matrix (Mat) options:
6  -mat_block_size <now -1 : formerly -1>: Set the blocksize used to store the matrix (MatSetBlockSize)
7  -mat_type <now aij : formerly aij>: Matrix type (one of) mffd mpimaij seqaij maij mpikaij seqkaij
  kaij is shell composite mpaij seqaij mpaijperm seqaijperm mpaijsell seqaijsell seqaijcr1 mpaijcr1
  mpibaij seqbaij mpisbaij seqsbaij mpdense seqdense mpiadj scatter blockmat nest mpisell seqsell
  preallocator dummy constantdiagonal (MatSetType)
8  -----
9  Options for SEAIJ matrix:
0  -mat_no_unroll: <FALSE : FALSE> Do not optimize for inodes (slower) (None)
1  -mat_no_inode: <FALSE : FALSE> Do not optimize for inodes -slower- (None)
2  -mat_inode_limit <now 5 : formerly 5>: Do not use inodes larger then this value (None)
3  -mat_is_symmetric: Checks if mat is symmetric on MatAssemblyEnd() (MatIsSymmetric)
4  -mat_is_symmetric <0. : 0.>: Checks if mat is symmetric on MatAssemblyEnd() (MatIsSymmetric)
5  -mat_null_space_test: <FALSE : FALSE> Checks if provided null space is correct in MatAssemblyEnd() (
  MatSetNullSpaceTest)
6  -mat_error_if_failure: <FALSE : FALSE> Generate an error if an error occurs when factoring the matrix
  (MatsetErrorIfFailure)
```

(Refer Slide Time: 21:59)

```
3  -mat_is_symmetric: Checks if mat is symmetric on MatAssemblyEnd() (MatIsSymmetric)
4  -mat_is_symmetric <0. : 0.>: Checks if mat is symmetric on MatAssemblyEnd() (MatIsSymmetric)
5  -mat_null_space_test: <FALSE : FALSE> Checks if provided null space is correct in MatAssemblyEnd() (
  MatSetNullSpaceTest)
6  -mat_error_if_failure: <FALSE : FALSE> Generate an error if an error occurs when factoring the matrix
  (MatsetErrorIfFailure)
7  -mat_new_nonzero_location_err: <FALSE : FALSE> Generate an error if new nonzeros are created in the
  matrix structure (useful to test preallocation) (MatSetOption)
8  -mat_new_nonzero_allocation_err: <FALSE : FALSE> Generate an error if new nonzeros are allocated in
  the matrix structure (useful to test preallocation) (MatSetOption)
9  -----
0  Viewer (-mat_view) options:
1  -mat_view ascii[:filename][:format][:append]]: Prints object to stdout or ASCII file (
  PetscOptionsGetViewer)
2  -mat_view binary[:filename][:format][:append]]: Saves object to a binary file (
  PetscOptionsGetViewer)
3  -mat_view draw[:drawtype][:filename|format]] Draws object (PetscOptionsGetViewer)
4  -mat_view socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
5  -mat_view saws[:communicatorname]: Publishes object to SAMs (PetscOptionsGetViewer)
6  -----
7  Preconditioner (PC) options:
8  -pc_type <now ilu : formerly ilu>: Preconditioner (one of) none jacobi pbjacobi vpbjacobi bjacobi sor
  lu shell mg eisenstat ilu icc cholesky asm gasm ksp composite redundant nn mat fieldsplit galerkin
  exotic cp isc redistribute svd gang kaczmazr telescope patch hmg tfs bddc lmvn deflation (PCSetType)
```

So, what are the different. So, it shows you the different vec view options it shows the different matrix options.

(Refer Slide Time: 22:03)

```
1 -mat_view ascii[:filename][:format][:append]]: Prints object to stdout or ASCII file (
  PetscOptionsGetViewer)
2 -mat_view binary[:filename][:format][:append]]: Saves object to a binary file (
  PetscOptionsGetViewer)
3 -mat_view draw[:drawtype][:filename][format]] Draws object (PetscOptionsGetViewer)
4 -mat_view socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
5 -mat_view saws[:communicatorname]: Publishes object to SAs (PetscOptionsGetViewer)
6 -----
7 Preconditioner (PC) options:
8 -pc_type <now ilu : formerly ilu>: Preconditioner (one of) none jacobi pbjacobi vpbjacobi bjacobi sor
  lu shell mg eisenstat ilu icc cholesky asm gasm ksp composite redundant nn mat fieldsplit galerkin
  exotic cp lsc redistribute svd gang kacmarz telescope patch hmg tfs bddc lvmv deflation (PCSetType)
9 -pc_use amat: <FALSE : FALSE> use Amat (instead of Pmat) to define preconditioner in nested inner
  solves (PCSetUseamat)
10 ILU Options
11 -pc_factor_in_place: <FALSE : FALSE> Form factored matrix in the same memory as the matrix (
  PCFactorSetUseInPlace)
12 -pc_factor_fill <1. : 1.>: Expected non-zeros in factored matrix (PCFactorSetFill)
13 -pc_factor_shift_type <now NONE : formerly NONE> Type of shift to add to diagonal (choose one of) NONE
  NONZERO POSITIVE DEFINITE INBLOCKS (PCFactorSetShiftType)
14 -pc_factor_shift_amount <2.22045e-14 : 2.22045e-14>: Shift added to diagonal (PCFactorSetShiftAmount)
15 -pc_factor_zeropivot <2.22045e-14 : 2.22045e-14>: Pivot is considered zero if less than (
  PCFactorSetZeroPivot)
16 -pc_factor_column_pivot <-2. : -2.>: Column pivot tolerance (used only for some factorization) (
```

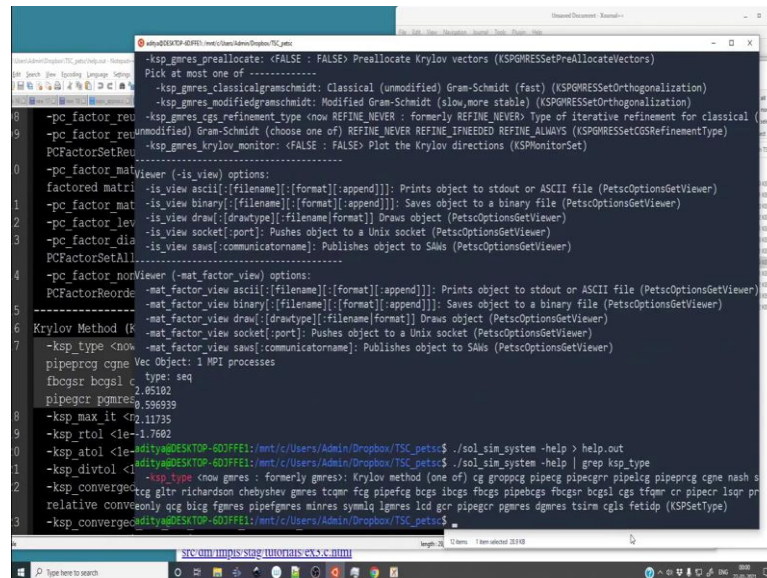
It shows the different viewer options, the different preconditioner options. So, ilu then you have jacobi, pbjacobi, vpbjacobi, bjacobi, sor, shell, mg, eisenstat, ilu, icc, cholesky, asm. So, different kinds of preconditioners are present. So, we are more interested in the KSP options.

(Refer Slide Time: 22:24)

```
8 -pc_factor_reuse_fill: <FALSE : FALSE> Use fill from previous factorization (PCFactorSetReuseFill)
9 -pc_factor_reuse_ordering: <FALSE : FALSE> Reuse ordering from previous factorization (
  PCFactorSetReuseOrdering)
10 -pc_factor_mat_ordering_type <now natural : formerly natural>: Reordering to reduce nonzeros in
  factored matrix (one of) natural nd lwd rcm gmd rowlength spectral (PCFactorSetMatOrderingType)
11 -pc_factor_mat_solver_type <now (null) : formerly (null)>: Specific direct solver to use (MatGetFactor)
12 -pc_factor_levels <now 0 : formerly 0>: levels of fill (PCFactorSetLevels)
13 -pc_factor_diagonal_fill: <FALSE : FALSE> Allow fill into empty diagonal entry (
  PCFactorSetAllowDiagonalFill)
14 -pc_factor_nonzeros_along_diagonal: Reorder to remove zeros from diagonal (
  PCFactorReorderForNonzeroDiagonal)
15 -----
16 Krylov Method (KSP) options:
17 -ksp_type <now gres : formerly gres>: Krylov method (one of) cg groppcg pipecg pipecgr pipecg
  pipepcg cgm nash stcg gltr richardson chebyshev gres tcgrm fcg pipefcg bcgs ibcgs fbcgs pipebcgs
  fbcgr bcgs1 cgs tfqmr cr pipecr lsqr preonly gcg bicg fgmres pipefgmres minres symmlq lgmres lod gcr
  pipegr pgmres dgmres tsirm cglis fetidp (KSPSetType)
18 -ksp_max_it <now 10000 : formerly 10000>: Maximum number of iterations (KSPSetTolerances)
19 -ksp_rtol <1e-05 : 1e-05>: Relative decrease in residual norm (KSPSetTolerances)
20 -ksp_atol <1e-50 : 1e-50>: Absolute value of residual norm (KSPSetTolerances)
21 -ksp_divtol <10000. : 10000.>: Residual norm increase cause divergence (KSPSetTolerances)
22 -ksp_converged_use_initial_residual_norm: <FALSE : FALSE> Use initial residual norm for computing
  relative convergence (KSPConvergedDefaultSetUITNorm)
23 -ksp_converged_use_min_initial_residual_norm: <FALSE : FALSE> Use minimum of initial residual norm and
```

So, there are a bunch of options and we want to extract this line. So, how do you extract this line?

(Refer Slide Time: 22:40)



```
ksp_help
-ksp_gmres_preallocate: <FALSE : FALSE> Preallocate Krylov vectors (KSPGMRESSetPreAllocateVectors)
Pick at most one of -----
-ksp_gmres_classicalgramschmidt: Classical (unmodified) Gram-Schmidt (fast) (KSPGMRESSetOrthogonalization)
-ksp_gmres_modifiedgramschmidt: Modified Gram-Schmidt (slow,more stable) (KSPGMRESSetOrthogonalization)
8 -pc_factor_reu -ksp_gmres_refinement_type <now REFINE_NEVER : formerly REFINE_NEVER> Type of iterative refinement for classical (KSPGMRESSetRefinementType)
9 -pc_factor_reu (unmodified) Gram-Schmidt (choose one of) REFINE_NEVER REFINE_IPNEEDED REFINE_ALWAYS (KSPGMRESSetCGSRefinementType)
PCFactorSetReu -ksp_gmres_krylov_monitor: <FALSE : FALSE> Plot the Krylov directions (KSPMonitorSet)
-----
0 -pc_factor_mat viewer (-is_view) options:
factored matrl -is_view ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
1 -pc_factor_mat -is_view binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
2 -pc_factor_mat -is_view draw[:drawtype][:filename[:format]]] Draws object (PetscOptionsGetViewer)
3 -pc_factor_mat -is_view socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
PCFactorSetAll -is_view saws[:communicatorname]: Publishes object to SAs (PetscOptionsGetViewer)
-----
4 -pc_factor_mat viewer (-mat_factor_view) options:
PCFactorReorde -mat_factor_view ascii[:filename][:format][:append]]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
5 -mat_factor_view binary[:filename][:format][:append]]]: Saves object to a binary file (PetscOptionsGetViewer)
6 -mat_factor_view draw[:drawtype][:filename[:format]]] Draws object (PetscOptionsGetViewer)
7 Krylov Method (KSP) -mat_factor_view socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
8 -mat_factor_view saws[:communicatorname]: Publishes object to SAs (PetscOptionsGetViewer)
9 -ksp_type <now pipeprog cgne Vec Object: 1 MPI processes
pipeprog cgne type: seq
fgcgst bcgsl c type: seq
2.05102
pipepgr pgmres 0.596939
8 -ksp_max_it <2.11735
9 -ksp_rtol <1e-1.7692
-ksp_atol <1e-14.7692
-ksp_divtol <1
-ksp_type <now gmres : formerly gmres> Krylov method (one of) cg groppcg pipecg pipecgr pipepgr pipepgr cgne nash s
-ksp_convergecg gtr richardson chebyshev gmres tqm frg pipefcg bcgs lbcs fgcs pipebcgs fbcgsr bcgsl cgs tfqm cr pipecr lqgr pr
relative convergeonly qcg bicg fgmes pipefgmes minres symmlq lgmres lcg ger pipepgr pgmres dgmes tsirm cglis fetidp (KSPSetType)
-ksp_convergecg
81tya@DESKTOP-60JFFE1:~/nt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, there are in linux there are specialized tools one of the specialized tools is called as grep. So, it will scan. So, we are piping this help output to grep which will give us the line which contains the keyword. So, the keyword will be the keyword we want is ksp type.

So, we are going to give it ksp type. So, the output of grep is shown over here. So, grep is a linux tool you do not need to worry about it. So, one of these methods cg, groppcg, pipecg, pipecgr the host of methods minres, fgmres, biconjugate, gradients stuff like that. So, you have a whole set of methods that very efficient and intelligent people have coded so that you do not have to go ahead and code all this.

But remember novel it always helps to know what each algorithm does inside out, but it is unlikely that someone like me is going to write a code which is multi-processor which is more efficient than what these people have it is unlikely.

(Refer Slide Time: 24:02)

```

-Is_view_ascii[:filename][:format][:append]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
-Is_view_binary[:filename][:format][:append]]: Saves object to a binary file (PetscOptionsGetViewer)
-Is_view_draw[:drawtype][:filename[:format]]: Draws object (PetscOptionsGetViewer)
-Is_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-Is_view_saws[:communicatorname]: Publishes object to SAs (PetscOptionsGetViewer)

-pc_factor_reu
-pc_factor_reu
Viewer (-mat_factor_view) options:
PCFactorSetReu
-pc_factor_mat
-mat_factor_view_ascii[:filename][:format][:append]]: Prints object to stdout or ASCII file (PetscOptionsGetViewer)
factored mat
-mat_factor_view_binary[:filename][:format][:append]]: Saves object to a binary file (PetscOptionsGetViewer)
-pc_factor_mat
-mat_factor_view_draw[:drawtype][:filename[:format]]: Draws object (PetscOptionsGetViewer)
-pc_factor_mat
-mat_factor_view_socket[:port]: Pushes object to a Unix socket (PetscOptionsGetViewer)
-pc_factor_level
-mat_factor_view_saws[:communicatorname]: Publishes object to SAs (PetscOptionsGetViewer)
-pc_factor_dio
Vec Object: 1 MPI processes
type: seq
PCFactorSetAll
1.05102
-pc_factor_norm
0.596939
PCFactorReorder
2.11735
-1.7602

Krylov Method
Krylov Method (one of) cg grompp pipecg pipecgrn pipekcg pipekprc gmres nash s
-pc_type <now
-ksp_type <now gmres : formerly gmres: Krylov method (one of) cg grompp pipecg pipecgrn pipekcg pipekprc gmres nash s
pipeprcg gmres
tgf gltr richardson chebyshev gmres tcqm fcg pipefcg bcgs lbcs fcgs pipebcgs fbcgsr bcgsl cgs tfqm cr pipecr lsqr pr
fbcgsr bcgsl
only qcg bicg fgmes pipefgmes minres symmlq lgmes lcg gor pipegor gmres dgmres tsirm cglis fetidp (KSPSetType)
pipeprc gmres
0 KSP Residual norm 3.484855562939e+00
-pc_max_it <0
1 KSP Residual norm 2.645396684416e-15
-ksp_rtol <le-Vec Object: 1 MPI processes
type: seq
-ksp_atol <le-
2.05102
-ksp_divtol <
0.596939
-ksp_converged
2.11735
relative converge
-1.7602
-ksp_converged

```

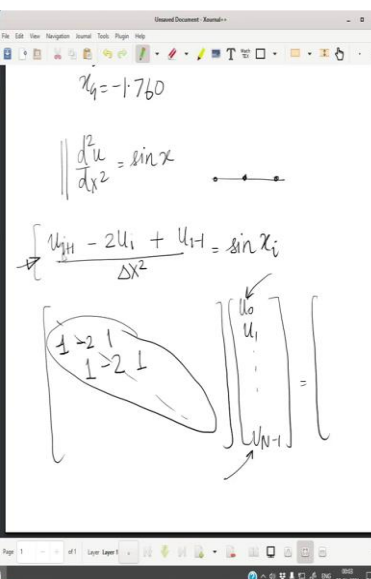
So in fact, let us change some more options. So, now, that we know these methods exist we can call minres. So, it uses the minres thing it gives us a slightly different norm after one iteration, but it has already converged you do not need to worry about it. So, that is how you can choose solvers at runtime I have shown you a few linux tips and tricks. So, now, let us solve for a tri-diagonal system.

(Refer Slide Time: 24:48)

```

1 #include <petsc.h>
2
3 int main(int argc, char **argv)
4 {
5     Mat A;
6     Vec b, x;
7     KSP ksp;
8     int i;
9     int j[4] = {0, 1, 2, 3};
10    double ab[4] = {7.0, 1.0, 2.0, 4.0};
11    double aA[4][4] = {{1, 0, 4, 2},
12                      {2, 6, 1, 5},
13                      {0, 1, -1, -2},
14                      {4, 3, -2, 1}};
15
16    PetscInitialize(&argc, &argv, NULL, "Create vector AND
17
18    VecCreate(PETSC_COMM_WORLD, &b);
19    VecSetSizes(b, PETSC_DECIDE, 4);
20    VecSetFromOptions(b);
21    VecSetValues(b, 4, j, ab, INSERT_VALUES);
22    VecAssemblyBegin(b); VecAssemblyEnd(b);

```



So, what I am trying to get at is if you are solving a boundary value problem in 1D say $d^2u/dx^2 = \sin x$ right if you are trying to solve this. So, what will you do? You are going

to split this into a discrete form over one central node one right node and one left node. So, what do you have? You have $u_{i+1} - 2u_i + u_{i-1}$ upon $\Delta x^2 = \sin x_i$.

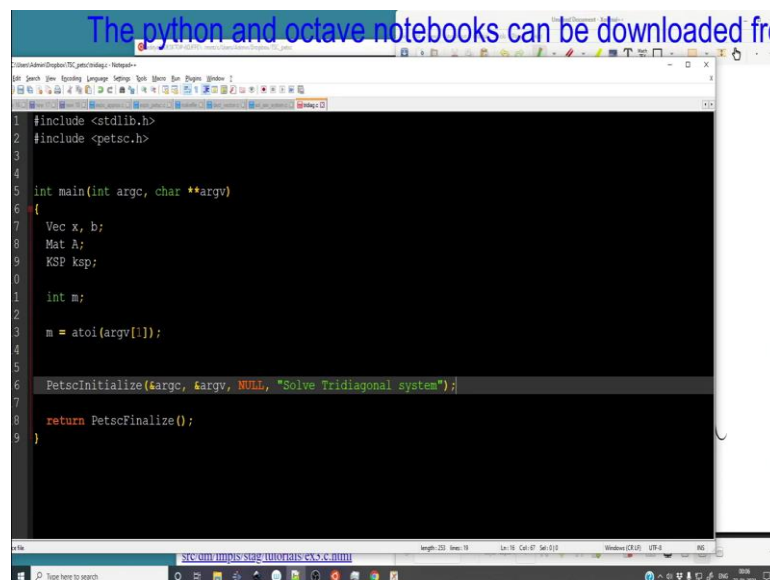
And then as we have seen in the python code you assemble this matrix when assembling this matrix apart from the boundaries you get a tri-diagonal system ok. So, the diagonal elements will be - 2 the off diagonal will be 1 the left angle will be 1 it will keep doing this except the boundary.

Because at the boundary the value of u will be specified or the gradient will be specified after which you can do, you can take a ghost node and you can find out the appropriate boundary condition using this equation itself and we have covered this how a tri diagonal system comes into existence.

So, this kind of tri diagonal system has to be often times algorithmically done I mean in this particular example it is still very easy, but nevertheless you have to build it for an arbitrary size we do not restrict it to a 4 by 4 matrix. So, we must write the code so that you can assemble the matrix for any arbitrary size. So, let us see how that can be done.

(Refer Slide Time: 26:51)

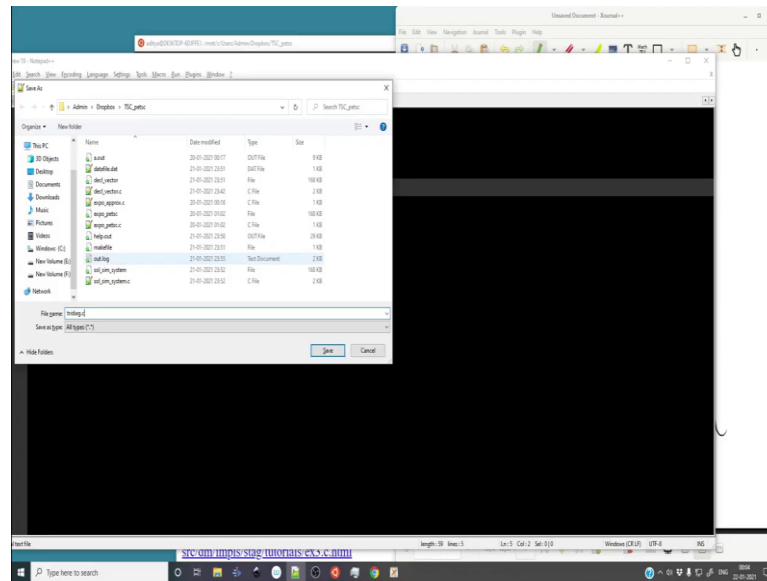
The python and octave notebooks can be downloaded from



```
1 #include <stdlib.h>
2 #include <petsc.h>
3
4
5 int main(int argc, char **argv)
6 {
7     Vec x, b;
8     Mat A;
9     KSP ksp;
10
11     int m;
12     m = atoi(argv[1]);
13
14     PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
15
16     return PetscFinalize();
17 }
18
```

So, first of all we are going to have hash include std not std petsc dot h int main int argc char start star argv.

(Refer Slide Time: 27:14)



So, let us save it as tridiag dot c and we will return let us see finalize alright. So, we are going to define the vectors as we did in the last code x and b we are going to define the matrix A we are going to define the KSP object ksp and what we are going to do now is define the integer m what will m do? It will hold the size of the matrix and we can put some default or we can overload it with the help of some command line argument.

Let us keep it m for now and let us write m equal to atoi of argv first thing. So, we must pass through the command line argument the integer and the size of the matrix. So, let us include stdlib as well just in case let me put this before this ok. So, m contains this and we will define all the other parameters as we go in the code.

So, first things first we must do a PetscInitialize this should contain PETSC COMM WORLD and no this should not contain PETSC COMM WORLD rather it should contain the address of argc address of argv and null and some help string alright.

(Refer Slide Time: 29:11)

The python and octave notebooks can be downloaded from http://www.facweb.iitkgp.ac.in/~adityab/lecture_list.html

```
1 #include <stdlib.h>
2 #include <petsc.h>
3
4
5 int main(int argc, char **argv)
6 {
7     Vec x, b;
8     Mat A;
9     KSP ksp;
10
11     int m;
12
13     m = atoi(argv[1]);
14
15     PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
16
17     return PetscFinalize();
18 }
19
```

(Refer Slide Time: 29:30)

[/www.facweb.iitkgp.ac.in/~adityab/lecture_list.html](http://www.facweb.iitkgp.ac.in/~adityab/lecture_list.html) as a quick reference

```
1 #include <stdlib.h>
2 #include <petsc.h>
3
4
5 int main(int argc, char **argv)
6 {
7     Vec x, b;
8     Mat A;
9     KSP ksp;
10
11     int m;
12
13     m = atoi(argv[1]);
14
15     PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
16
17     VecCreate(PETSC_COMM_WORLD, &b);
18     VecSetSizes(b, PETSC_DECIDE, 4);
19     VecSetFromOptions(b);
20     VecSetValues(b, 4, j, ab, INSERT_VALUES);
21     VecAssemblyBegin(b); VecAssemblyEnd(b);
22
23     return PetscFinalize();
24 }
25
```

So, now we have to do the same set of lines and which we had done well we had not done, but we need to declare this vector.

(Refer Slide Time: 29:42)

```
wnloaded from http://www.facweb.iitkgp.ac.in/~adityab/lecture_1

double ab[4] = {7.0, 1.0, 2.0, 4.0};
double aA[4][4] = {{1, 0, 4, 2},
                  {2, 6, 1, 5},
                  {0, 1, -1, -2},
                  {4, 3, -2, 1}};

PetscInitialize(&argc, &argv, NULL, "Create vector AND matrix\n");

VecCreate(PETSC_COMM_WORLD, &b);
VecSetSizes(b, PETSC_DECIDE, 0);
VecSetFromOptions(b);
VecSetValues(b, 4, j, ab, INSERT_VALUES);
VecAssemblyBegin(b); VecAssemblyEnd(b);

MatCreate(PETSC_COMM_WORLD, &A);
MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, 4, 4);
MatSetFromOptions(A);
MatSetOp(A);
for (i = 0; i < 4; i++)
{
  MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
}
```

So, let me copy this, let us not waste time in writing all this we have to create the vector.

(Refer Slide Time: 29:48)

```
Vec x, b, xexact;
Mat A;
KSP ksp;

int m;
m = atoi(argv[1]);

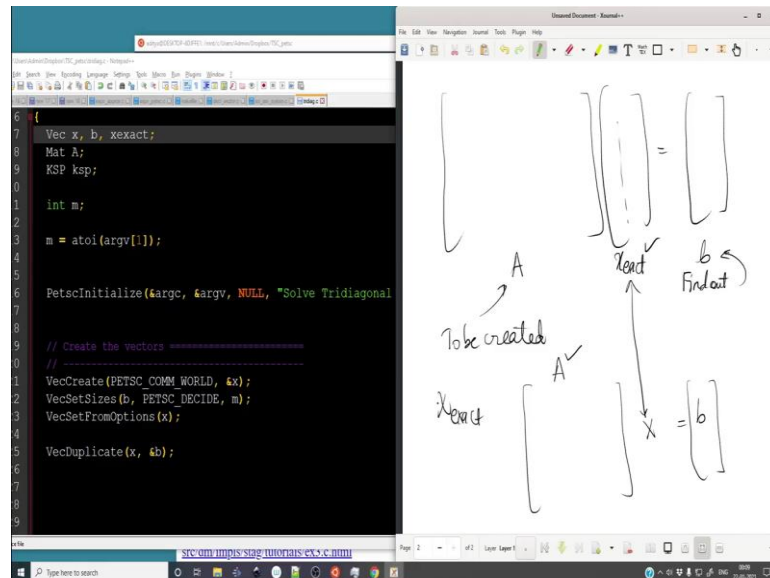
PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");

// Create the vectors
VecCreate(PETSC_COMM_WORLD, &x);
VecSetSizes(b, PETSC_DECIDE, m);
VecSetFromOptions(x);
VecDuplicate(x, &b);
```

So, this bit of code is to create the vectors it is good to always type down some comments meaningful comments. So, what kind of vector do we want to build? So, we want to build let us create x and the size will be m it will not be 4 alright it will be m. So, we will build. So, set values we have to get rid of because we are going to do it somewhere else alright; let me keep it we do not need and we can delete it for now.

So, we just do set values of options once x is created we can also create b. So, we can do VecDuplicate. So, we are going to pass x and the value of b. So, essentially we have created duplicate vector b alright then. In fact, we are going to make a very synthetic problem. So, we will need something called as x exact. So, what we are going to do is quite simple.

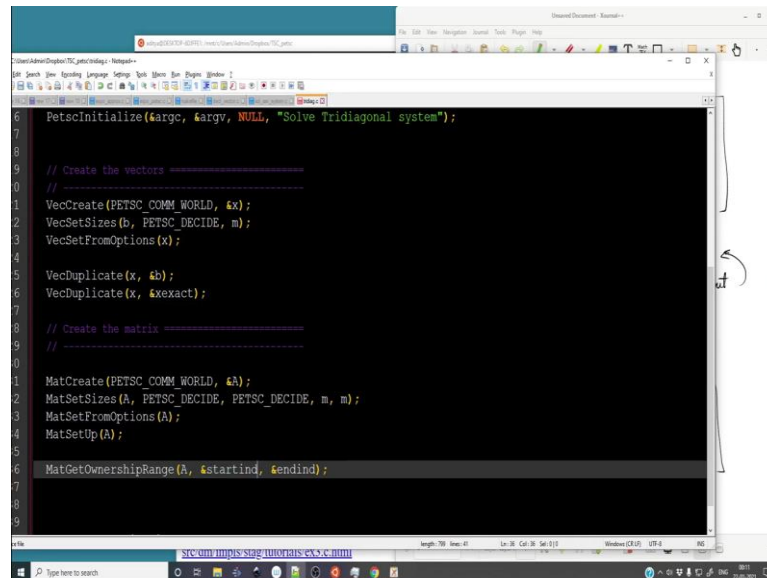
(Refer Slide Time: 31:15)



We are going to take our differentiation matrix A. So, we are going to create this A to be created we are going to create x exact that is we will populate this with something which is known ok and then we are going to multiply these two and find out some matrix b once we have b then we are going to synthetically solve again.

So, A is known X will now be unknown equal to b. So, we are going to solve this afterwards and we are going to compare x exact and X well in the ideal world they should be exactly equal, but let us see because we are using iterative algorithms we will have large matrices and they may be close, but definitely not equal and that is; and that is the thing you have to live with numerical algorithms.

(Refer Slide Time: 32:22)



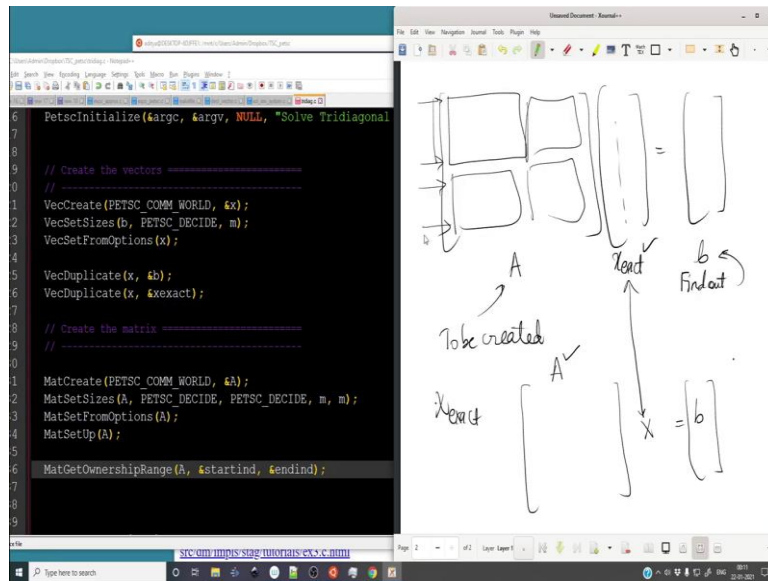
```
6 PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
7
8
9 // Create the vectors -----
10 // -----
11 VecCreate(PETSC_COMM_WORLD, &x);
12 VecSetSizes(b, PETSC_DECIDE, m);
13 VecSetFromOptions(x);
14
15 VecDuplicate(x, &b);
16 VecDuplicate(x, &xexact);
17
18 // Create the matrix -----
19 // -----
20
21 MatCreate(PETSC_COMM_WORLD, &A);
22 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
23 MatSetFromOptions(A);
24 MatSetUp(A);
25
26 MatGetOwnershipRange(A, &startind, &endind);
27
28
29
```

So, set from options x we are going to duplicate we are going to make another duplicate of x comma and there is an xexact alright. So, far we have created the three matrices now we are going to create matrix. So, again if we going to reuse some of the code we are going to reuse this.

So, we have to create the matrix A you have to set sizes instead of 4 it will be m right then MatrixSetFromOptions A good what else and we have to do matrix MatsSetup A something which we have done over here as well. So, once we have this we have now created the two vectors another three vectors and the matrix.

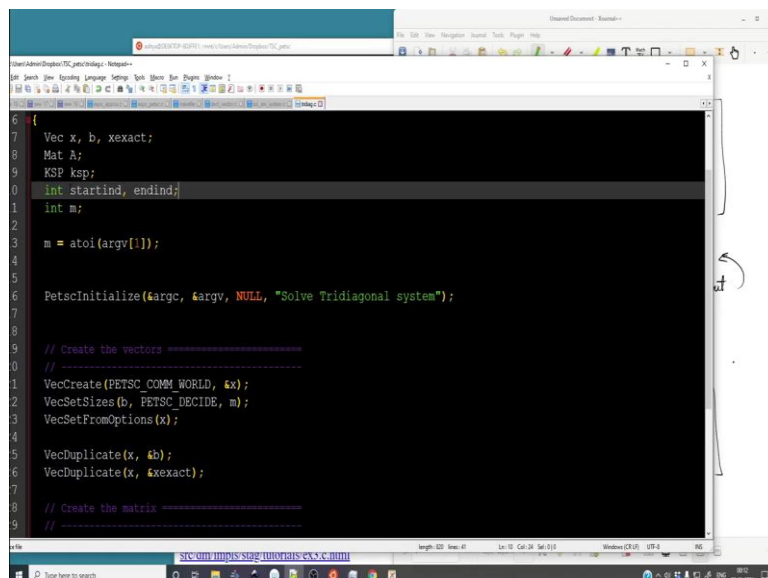
So, what we are going to do is we are going to first obtain the loop range. So, let me show you we are going to do MatGetOwnershipRange A this will be startind this will be endind this means it will look at. So, the program will look at the array A if you have multiple processes it will distribute the index over which the program is going to loop it will divide into sub chunks ok.

(Refer Slide Time: 34:24)



So, it can have something like this startind is this endind is this startind is this it depends if you are doing it in parallel I am not simply speaking startind and endind will simply contain the dimension of the array. So, hence the ownership range different processors will own different ranges and what we can do already is so we have pretend declare startind and endind because if we have not declared them. So, they need to be declared as integers.

(Refer Slide Time: 35:07)



So, let us do that. So, over here we need to make int startind and endind. So, we have created the startind and endind. So in fact, we can print it out and I will show you before doing all these let us print out certain parts and we will see what we see. So, let me destroy the different things. So, let me reuse some of the code to destroy we will need all of this.

(Refer Slide Time: 35:35)

```

1 {
2     MatSetValues(A, 1, &i, 4, j, aA[i], INSERT_VALUES);
3 }
4 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
5
6 //MatView(A, PETSC_VIEWER_STDOUT_WORLD);
7 //VecView(b, PETSC_VIEWER_STDOUT_WORLD);
8
9 VecDuplicate(b, &x);
10
11 KSPCreate(PETSC_COMM_WORLD, &ksp);
12 KSPSetOperators(ksp, A, A);
13 KSPSetFromOptions(ksp);
14 KSPSolve(ksp, b, x);
15
16 VecView(x, PETSC_VIEWER_STDOUT_WORLD);
17
18 MatDestroy(&A);
19 VecDestroy(&b);
20 VecDestroy(&x);
21 KSPDestroy(&ksp);
22 return PetscFinalize();
23 }

```

(Refer Slide Time: 35:39)

```

1 VecCreate(PETSC_COMM_WORLD, &x);
2 VecSetSizes(b, PETSC_DECIDE, m);
3 VecSetFromOptions(x);
4
5 VecDuplicate(x, &b);
6 VecDuplicate(x, &xexact);
7
8 // Create the matrix =====
9 // =====
10
11 MatCreate(PETSC_COMM_WORLD, &A);
12 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
13 MatSetFromOptions(A);
14 MatSetUp(A);
15
16 MatGetOwnershipRange(A, &startind, &endind);
17
18 MatDestroy(&A);
19 VecDestroy(&b);
20 VecDestroy(&x); VecDestroy(&xexact);
21 KSPDestroy(&ksp);
22 return PetscFinalize();
23 }

```

So, destroy a b x you have to destroy xexact as well. So, let me modify the make file to have a new target alright.

(Refer Slide Time: 35:51)

```
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14   $(RM) sol_sim_system.o
15
16 sol_sim_system: sol_sim_system.o chkopts
17   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
18   $(RM) sol_sim_system.o
```

(Refer Slide Time: 36:03)

```
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14   $(RM) sol_sim_system.o
15
16 tridiag: tridiag.o chkopts
17   -$(LINKER) -o tridiag tridiag.o $(PETSC_LIB)
18   $(RM) sol_sim_system.o
```

(Refer Slide Time: 36:09)

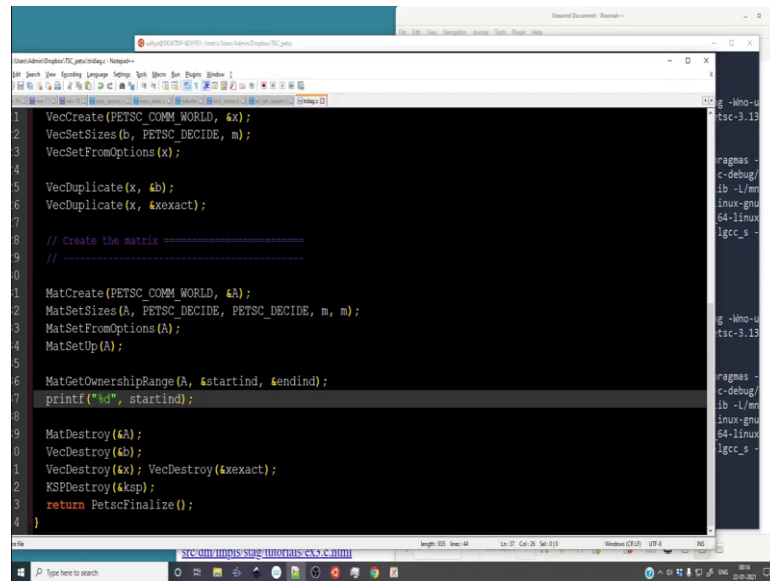
```
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14   $(RM) sol_sim_system.o
15
16 tridiag: tridiag.o chkopts
17   -$(LINKER) -o tridiag tridiag.o $(PETSC_LIB)
18   $(RM) tridiag.o
```

(Refer Slide Time: 36:14)

```
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./sol_sim_system -help > help.out
1 include $(PETSC_DIR)/lib/petsc/conf/variables
2 include $(PETSC_DIR)/lib/petsc/conf/rules
3
4 expo_petsc: expo_petsc.o chkopts
5   -$(LINKER) -o expo_petsc expo_petsc.o $(PETSC_LIB)
6   $(RM) expo_petsc.o
7
8 decl_vector: decl_vector.o chkopts
9   -$(LINKER) -o decl_vector decl_vector.o $(PETSC_LIB)
10  $(RM) decl_vector.o
11
12 sol_sim_system: sol_sim_system.o chkopts
13   -$(LINKER) -o sol_sim_system sol_sim_system.o $(PETSC_LIB)
14   $(RM) sol_sim_system.o
15
16 tridiag: tridiag.o chkopts
17   -$(LINKER) -o tridiag tridiag.o $(PETSC_LIB)
18   $(RM) tridiag.o
19
20 aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make tridiag
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 -o tridiag.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protector -fvissibility-hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -lmpi -rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -lm -rpath,/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 -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfbblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgcc_s -lquadmath -lstdc++ -ldl
/bin/rm -f tridiag.o
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag
Segmentation fault (core dumped)
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, we have created a new target. So, let me test it compiles well dot slash tridiag segmentation fault because you have not done anything most likely it has ran into something empty ok.

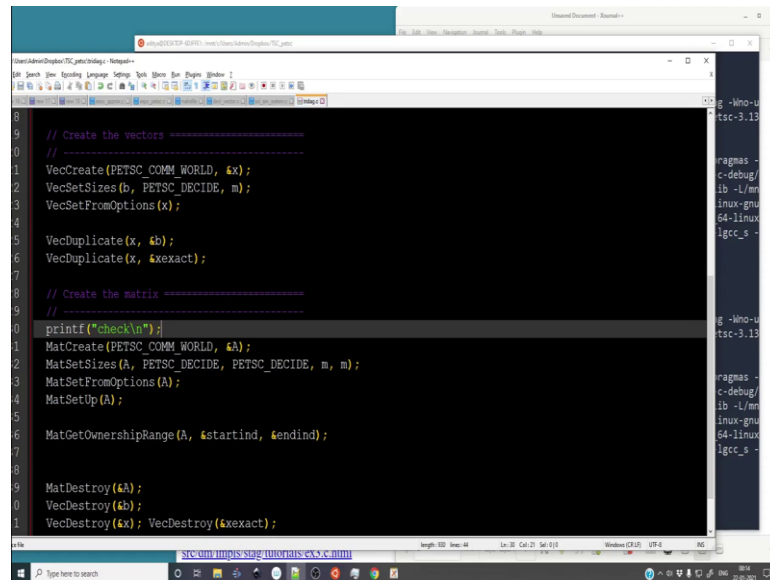
(Refer Slide Time: 36:36)



```
1 VecCreate(PETSC_COMM_WORLD, &x);
2 VecSetSizes(b, PETSC_DECIDE, m);
3 VecSetFromOptions(x);
4
5 VecDuplicate(x, &b);
6 VecDuplicate(x, &xexact);
7
8 // Create the matrix =====
9 // =====
10
11 MatCreate(PETSC_COMM_WORLD, &A);
12 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
13 MatSetFromOptions(A);
14 MatSetUp(A);
15
16 MatGetOwnershipRange(A, &startind, &endind);
17 printf("%d", startind);
18
19 MatDestroy(&A);
20 VecDestroy(&b);
21 VecDestroy(&x); VecDestroy(&xexact);
22 KSPDestroy(&ksp);
23 return PetscFinalize();
24 }
```

So, I do not know where the segmentation fault is, but let me see if I can print the startind or not the segmentation for much before this and that is ok. So, well you can actually pinpoint where the segmentation fault is most likely it is somewhere over here after setting up we have not done anything.

(Refer Slide Time: 37:12)



```
8
9 // Create the vectors =====
10 // =====
11 VecCreate(PETSC_COMM_WORLD, &x);
12 VecSetSizes(b, PETSC_DECIDE, m);
13 VecSetFromOptions(x);
14
15 VecDuplicate(x, &b);
16 VecDuplicate(x, &xexact);
17
18 // Create the matrix =====
19 // =====
20 printf("check\n");
21 MatCreate(PETSC_COMM_WORLD, &A);
22 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
23 MatSetFromOptions(A);
24 MatSetUp(A);
25
26 MatGetOwnershipRange(A, &startind, &endind);
27
28
29 MatDestroy(&A);
30 VecDestroy(&b);
31 VecDestroy(&x); VecDestroy(&xexact);
32 }
```

(Refer Slide Time: 37:24)

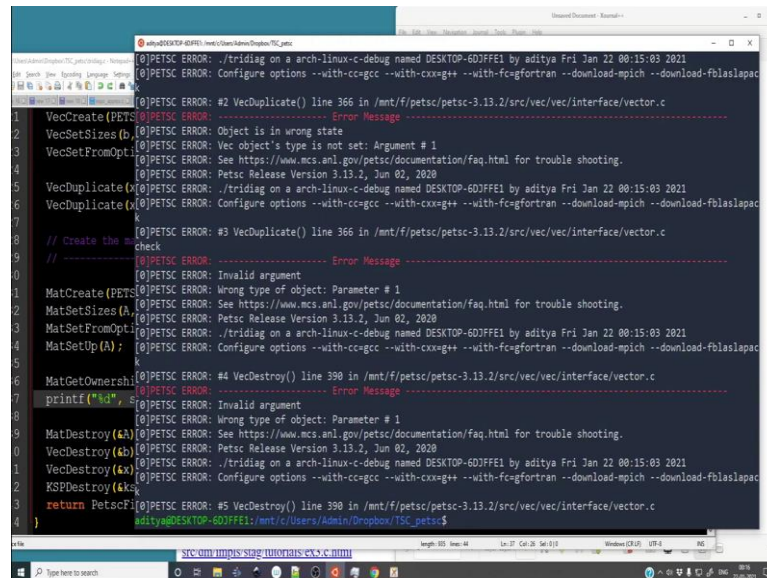
```
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag
Segmentation fault (core dumped)
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make tridiag
/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o tridiag.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protector -fvvisibility=hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include -I/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin
// Create the vector
// -----
Warning: chkopts target is deprecated and can be removed from user makefiles
VecCreate(PETSC_COMM_WORLD, &x);
VecSetSizes(b, PETSC_COMM_WORLD, m);
VecSetFromOptions(x);
VecDuplicate(x, &b);
VecDuplicate(x, &exact);
// Create the matrix -----
// -----
MatCreate(PETSC_COMM_WORLD, &A);
MatSetSizes(A, PETSC_COMM_WORLD, PETSC_COMM_WORLD, m, m);
MatSetFromOptions(A);
MatSetUp(A);
MatGetOwnershipRange(A, &startind, &endind);
printf("%d", startind);
MatDestroy(&A);
VecDestroy(&b);
VecDestroy(&exact);
KSPDestroy(&ksp);
return PetscFinalize();
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag
Segmentation fault (core dumped)
aditya@DESKTOP-6D3JFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

So, you can always do a printf to see I mean you can obviously use a debugger, but if you want to do it the old way then simply do this place it at various positions and see when it actually prints. It is not this we have not passed the number of not pass the elements. So, that is why m remains undefined and hence the segmentation fault well.

(Refer Slide Time: 37:45)

```
1 VecCreate(PETSC_COMM_WORLD, &x);
2 VecSetSizes(b, PETSC_COMM_WORLD, m);
3 VecSetFromOptions(x);
4
5 VecDuplicate(x, &b);
6 VecDuplicate(x, &exact);
7
8 // Create the matrix -----
9 // -----
10 MatCreate(PETSC_COMM_WORLD, &A);
11 MatSetSizes(A, PETSC_COMM_WORLD, PETSC_COMM_WORLD, m, m);
12 MatSetFromOptions(A);
13 MatSetUp(A);
14 MatGetOwnershipRange(A, &startind, &endind);
15 printf("%d", startind);
16 MatDestroy(&A);
17 VecDestroy(&b);
18 VecDestroy(&exact);
19 KSPDestroy(&ksp);
20 return PetscFinalize();
21 }
```

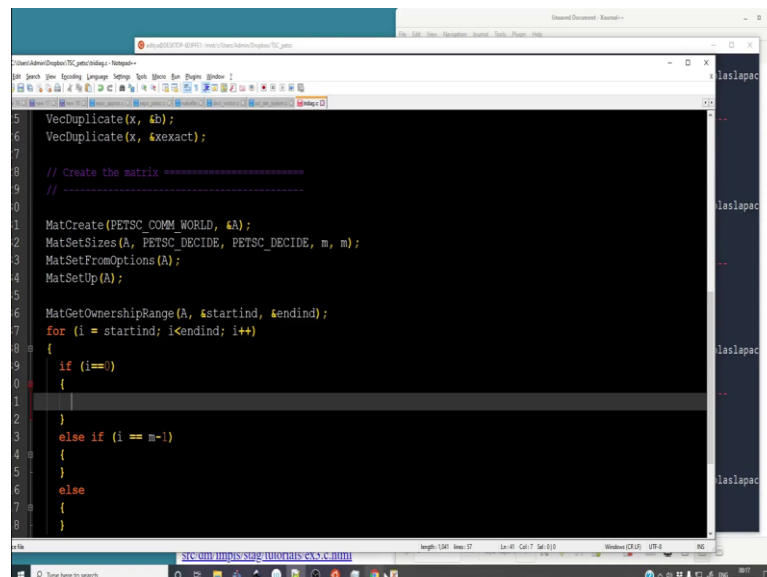

(Refer Slide Time: 37:48)



```
aditya@DESKTOP-6D3FFE1:~/c/Code/Advanced/vec_petsc$ ./tridiag on an arch-linux-c-debug named DESKTOP-6D3FFE1 by aditya Fri Jan 22 00:15:03 2021
[0]PETSC ERROR: Configure options --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --download-mpich --download-fblaslapack
[0]PETSC ERROR: #2 VecDuplicate() line 366 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/vector.c
----- Error Message -----
1 VecCreate(PETSC)
2 VecSetSizes(b,
3 VecSetFromOpt
4 VecDuplicate(0)
5 VecDuplicate(x)
6 // Create the m
7 // -----
8 [0]PETSC ERROR: Invalid argument
9 [0]PETSC ERROR: Wrong type of object: Parameter # 1
10 MatCreate(PETSC)
11 MatSetSizes(A,
12 MatSetFromOpt
13 MatSetUp(A);
14 MatGetOwnersh
15 printf("%d",
16 [0]PETSC ERROR: Invalid argument
17 [0]PETSC ERROR: Wrong type of object: Parameter # 1
18 MatDestroy(&b)
19 VecDestroy(&b)
20 VecDestroy(&a)
21 KSPDestroy(&k)
22 return PetscF
aditya@DESKTOP-6D3FFE1:~/c/Users/Admin/Dropbox/TSC_petsc$
```

So, let me put this we do have a big set of error and it appears to be because of the lack of declaration, can we scroll? Right. So, do not worry about these errors we will sort it out we will get rid of this. So, we have to get the membership range and what we will do is we will try to fill out the matrix now.

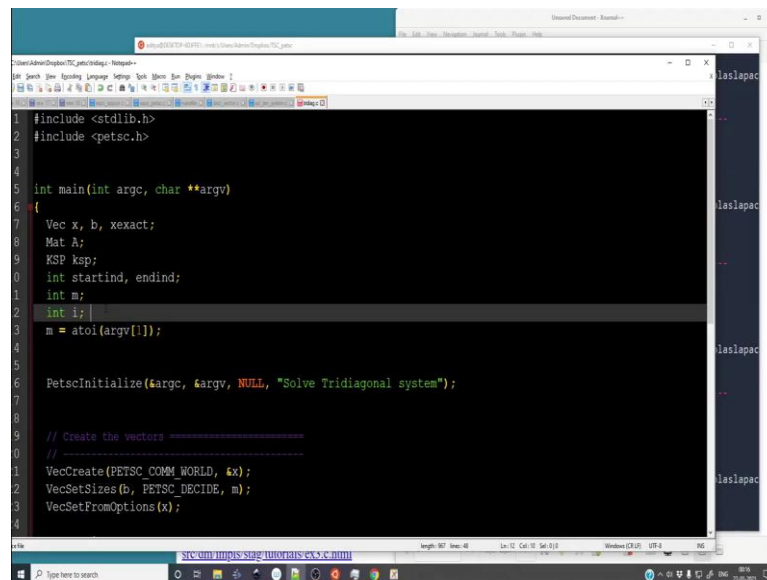
(Refer Slide Time: 39:14)



```
aditya@DESKTOP-6D3FFE1:~/c/Code/Advanced/vec_petsc$ ./tridiag on an arch-linux-c-debug named DESKTOP-6D3FFE1 by aditya Fri Jan 22 00:15:03 2021
[0]PETSC ERROR: Configure options --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --download-mpich --download-fblaslapack
[0]PETSC ERROR: #2 VecDuplicate() line 366 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/vector.c
----- Error Message -----
1 VecCreate(PETSC)
2 VecSetSizes(b,
3 VecSetFromOpt
4 VecDuplicate(0)
5 VecDuplicate(x)
6 // Create the m
7 // -----
8 [0]PETSC ERROR: Invalid argument
9 [0]PETSC ERROR: Wrong type of object: Parameter # 1
10 MatCreate(PETSC)
11 MatSetSizes(A,
12 MatSetFromOpt
13 MatSetUp(A);
14 MatGetOwnersh
15 printf("%d",
16 [0]PETSC ERROR: Invalid argument
17 [0]PETSC ERROR: Wrong type of object: Parameter # 1
18 MatDestroy(&b)
19 VecDestroy(&b)
20 VecDestroy(&a)
21 KSPDestroy(&k)
22 return PetscF
aditya@DESKTOP-6D3FFE1:~/c/Users/Admin/Dropbox/TSC_petsc$
```

So, for $i = \text{startind}$ $i < \text{endind}$ $i ++$. So, we are going to fill in all the elements but we need to define the counter i alright.

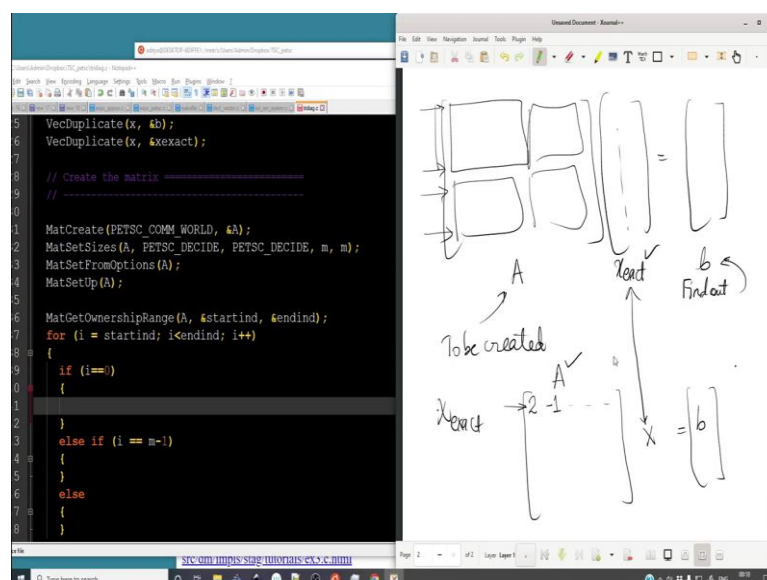
(Refer Slide Time: 39:35)



```
1 #include <stdlib.h>
2 #include <petsc.h>
3
4
5 int main(int argc, char **argv)
6 {
7     Vec x, b, xexact;
8     Mat A;
9     KSP ksp;
10    int startind, endind;
11    int m;
12    int i;
13    m = atoi(argv[1]);
14
15    PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
16
17    // Create the vectors -----
18    // -----
19    VecCreate(PETSC_COMM_WORLD, &x);
20    VecSetSizes(b, PETSC_DECIDE, m);
21    VecSetFromOptions(x);
22
```

So, over here we will do if $i = 0$, then we must do something else if $i = m - 1$ that is the end case we must do something else we must do something alright. So, what are these some things. So, if $i = 0$ it is like a boundary condition. So, we are going to set two elements. So, let us say we set two elements that is for $i = 0$ which is the first row we are going to set this to be 2 and this to be -1 everything else is 0.

(Refer Slide Time: 40:37)

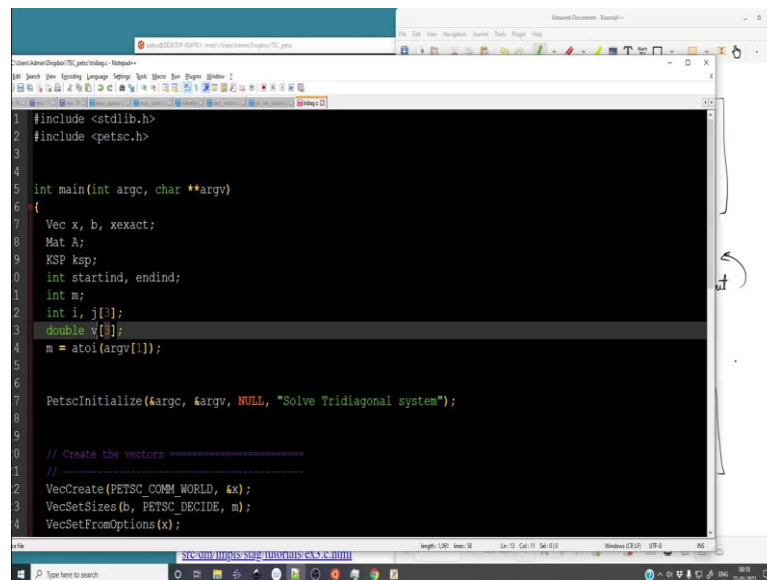


```
5 VecDuplicate(x, &b);
6 VecDuplicate(x, &xexact);
7
8 // Create the matrix -----
9 // -----
10
11 MatCreate(PETSC_COMM_WORLD, &A);
12 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
13 MatSetFromOptions(A);
14 MatSetUp(A);
15
16 MatGetOwnershipRange(A, &startind, &endind);
17 for (i = startind; i < endind; i++)
18 {
19     if (i == 0)
20     {
21     }
22 }
23 else if (i == m-1)
24 {
25 }
26 else
27 {
28 }
```

The diagram shows a matrix A and a vector b . The matrix A is represented as a grid of boxes. The first row of A is labeled "To be created" and contains the values 2, -1, and a series of dashes. The vector b is shown as a column of boxes, with the first element labeled "Final out". The equation $Ax = b$ is written below the diagram.

So, we must first create a vector which will sort of insert things into the matrix A. So, this is exactly in line with how we were inserting things.

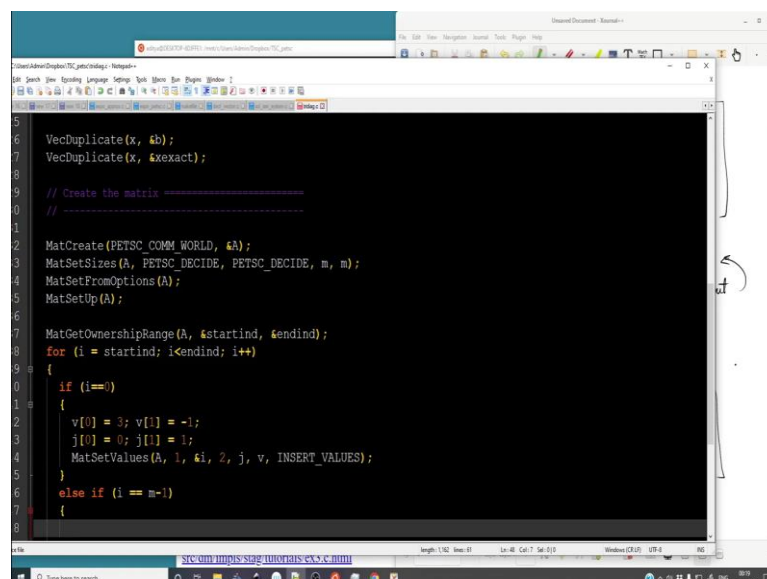
(Refer Slide Time: 41:00)



```
1 #include <stdlib.h>
2 #include <petsc.h>
3
4
5 int main(int argc, char **argv)
6 {
7     Vec x, b, xexact;
8     Mat A;
9     KSP ksp;
10    int startind, endind;
11    int m;
12    int i, j[3];
13    double v[3];
14    m = atoi(argv[1]);
15
16    PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
17
18    // Create the vectors -----
19    // -----
20    VecCreate(PETSC_COMM_WORLD, &x);
21    VecSetSizes(b, PETSC_DECIDE, m);
22    VecSetFromOptions(x);
```

So, we must define j as an array of 3 things. So, these 3 things we will define at runtime and we must define double b 3. So, we because at best we are going to insert 3 elements. So, when $i = 0$ then we must insert 3 and -1.

(Refer Slide Time: 41:31)



```
5
6 VecDuplicate(x, &b);
7 VecDuplicate(x, &xexact);
8
9 // Create the matrix -----
10 // -----
11
12 MatCreate(PETSC_COMM_WORLD, &A);
13 MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m, m);
14 MatSetFromOptions(A);
15 MatSetUp(A);
16
17 MatGetOwnershipRange(A, &startind, &endind);
18 for (i = startind; i < endind; i++)
19 {
20     if (i == 0)
21     {
22         v[0] = 3; v[1] = -1;
23         j[0] = 0; j[1] = 1;
24         MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
25     }
26     else if (i == m-1)
27     {
28
```

So, v_0 will be 3, v_1 will be -1. Similarly, j_0 that is the column index where we are going to insert something it is going to 0 and the column index were becoming sorry it should be j_1 where we are going to insert minus 1 has to be 1 essentially this location corresponds to j_0 this corresponding this location corresponds to $j = 1$.

So, now that we have this we going to simply MatSetValues; MatSetValues it will be A. So, we have to insert in A we are going to insert 1 row we are going to pass the row number we are going to insert 2 values we are going to insert at locations j we are going to insert the values of v and we will insert values that is it then if we have the last element we can simply reuse this code.

(Refer Slide Time: 42:51)

```

3     j[0] = 0; j[1] = 1;
4     MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
5 }
6 else if (i == m-1)
7 {
8     v[m-1] = 1;
9     j[0] = m-1;
10    MatSetValues(A, 1, &i, 1, j, v, INSERT_VALUES);
11 }
12 else
13 {
14     v[0] = 3; v[1] = -1;
15     j[0] = 0; j[1] = 1;
16    MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
17 }
18 }
19
20 MatDestroy(&A);
21 VecDestroy(&b);
22 VecDestroy(&x); VecDestroy(&xexact);
23 KSPDestroy(&ksp);
24 return PetscFinalize();
25 }

```

So, this will be b_{m-1} will be 3 and v_{m-2} or rather let us put it as a dirichlet condition. So, the first one is more akin to a Neumann Boundary Condition. So, simply v_{m-1} it will be equal to 1 and j_0 will be equal to $n-1$ this is one of the last value and we are going to insert only one value else what we are going to do is we are going to we are going to reuse this.

(Refer Slide Time: 44:04)

```
0  if (i==0)
1  {
2  v[0] = 2; v[1] = -1;
3  j[0] = 0; j[1] = 1;
4  MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
5  }
6  else if (i == m-1)
7  {
8  v[m-1] = 1;
9  j[0] = m-1;
10 MatSetValues(A, 1, &i, 1, j, v, INSERT_VALUES);
11 }
12 else
13 {
14 v[0] = -1; v[1] = 2;
15 j[0] = 0; j[1] = 1;
16 MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
17 }
18 }
19
20 MatDestroy(&a);
21 VecDestroy(&b);
22 VecDestroy(&x); VecDestroy(&exact);
```

So, v_0 will be -1 v_1 will be 2 this has to be 2 and v_2 will be -1. So, j_0 will be $i-1$ j_1 will be i j_2 will be $i+1$. So, we are going to insert 3 values this time so far so good we have not done something very remarkable, but we have just created the matrix A alright.

So, once we have made the matrix A let us proceed to set the exact solution as well. So, how do you do that? Well because we are doing this in a loop we can simply set values of the vector as well. So, we can say let us have the solution or something like x or let us have a periodic solution like something like $\sin i$ well it is periodic ok.

(Refer Slide Time: 45:39)

```
7 MatGetOwnershipRange(A, &startind, &endind);
8 for (i = startind; i < endind; i++)
9 {
10 if (i==0)
11 {
12 v[0] = 2; v[1] = -1;
13 j[0] = 0; j[1] = 1;
14 MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
15 }
16 else if (i == m-1)
17 {
18 v[m-1] = 1;
19 j[0] = m-1;
20 MatSetValues(A, 1, &i, 1, j, v, INSERT_VALUES);
21 }
22 else
23 {
24 v[0] = -1; v[1] = 2; v[2] = -1
25 j[0] = i-1; j[1] = i; j[2] = i+1;
26 MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
27 }
28 }
```

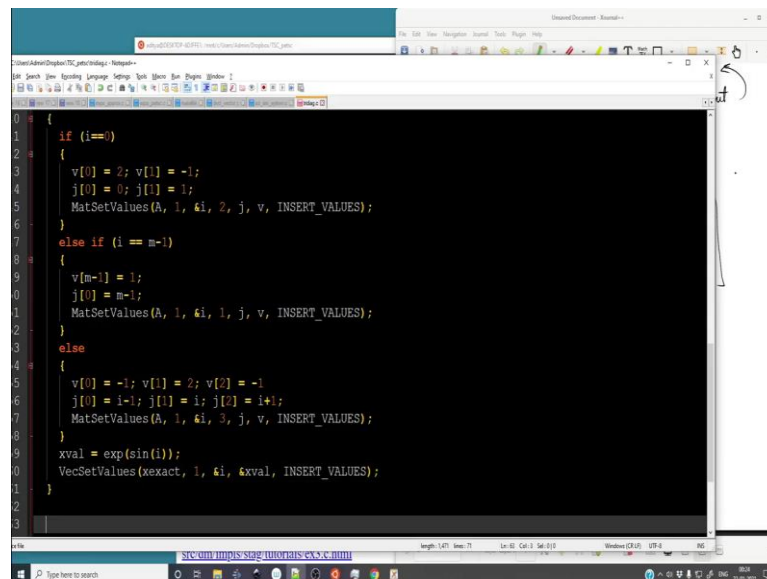
Handwritten notes on the whiteboard:

- Matrix A is shown as a block with arrows pointing to it from the text "To be created".
- Vector b is shown as a column vector $\begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix}$ with an arrow pointing to it from the text "Find it".
- Equation $Ax = b$ is written.
- The text "sin(i)" is written below the matrix.

Small video inset in the bottom right corner shows a person wearing a beanie and glasses.

So, let me. So, all these exact values will be simply sin of the row number.

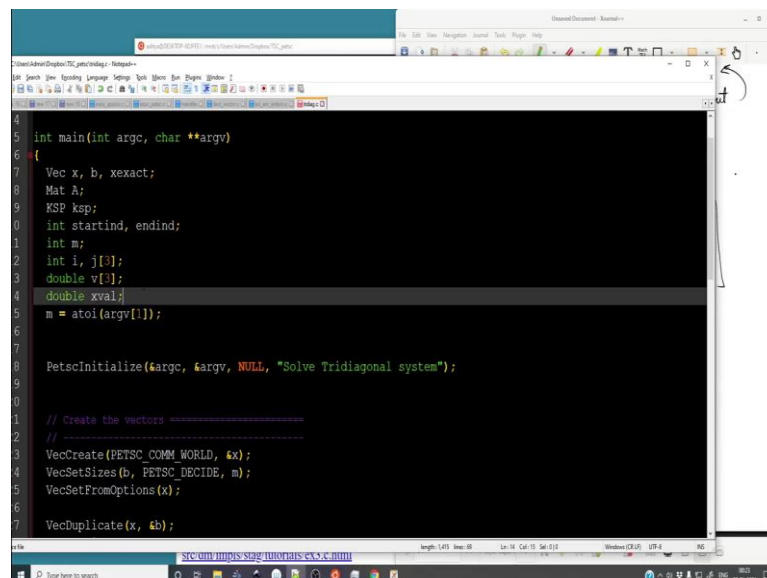
(Refer Slide Time: 45:56)



```
0 {
1   if (i==0)
2   {
3     v[0] = 2; v[1] = -1;
4     j[0] = 0; j[1] = 1;
5     MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
6   }
7   else if (i == m-1)
8   {
9     v[m-1] = 1;
10    j[0] = m-1;
11    MatSetValues(A, 1, &i, 1, j, v, INSERT_VALUES);
12  }
13  else
14  {
15    v[0] = -1; v[1] = 2; v[2] = -1
16    j[0] = i-1; j[1] = i; j[2] = i+1;
17    MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
18  }
19  xval = exp(sin(i));
20  VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
21 }
22 }
```

So, xval equal to. In fact, we can do an exponential of sin i.

(Refer Slide Time: 46:15)

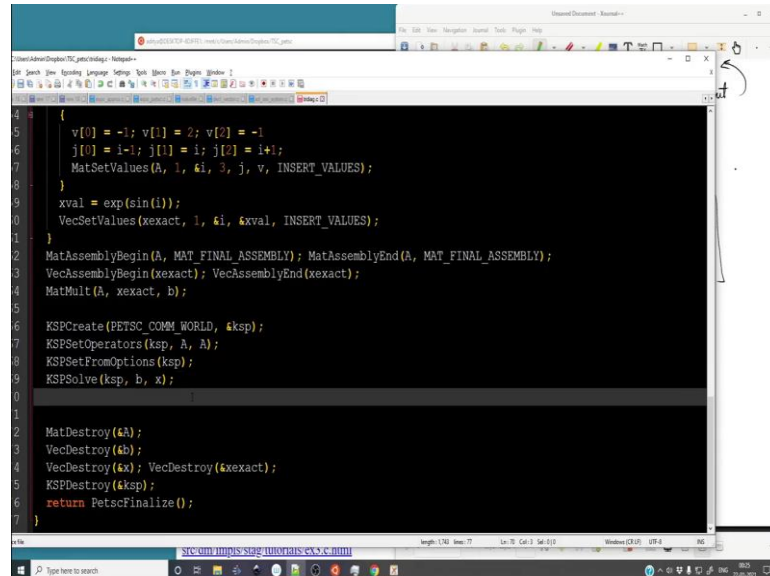


```
4
5
6
7
8 int main(int argc, char **argv)
9 {
10  Vec x, b, xexact;
11  Mat A;
12  KSP ksp;
13  int startind, endind;
14  int m;
15  int i, j[3];
16  double v[3];
17  double xval;
18  m = atoi(argv[1]);
19
20  PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
21
22  // Create the vectors -----
23  // -----
24  VecCreate(PETSC_COMM_WORLD, &x);
25  VecSetSizes(b, PETSC_DECIDE, m);
26  VecSetFromOptions(x);
27
28  VecDuplicate(x, &b);
29 }
```

So, we need to define xval. So, we need to define double xval what else once we have the xval we can simply VecSetValues xexact we are going to assign this value to xexact we are going to put 1 value we have to get the address of i we are going to insert we have to give the address of what we want to insert and finally, we are going to insert values ok. So, this is how we can put the exact solution as well.

So, now, with this we can do the matrix assembly. So, let us go to this code and let us copy this snippet.

(Refer Slide Time: 47:09)



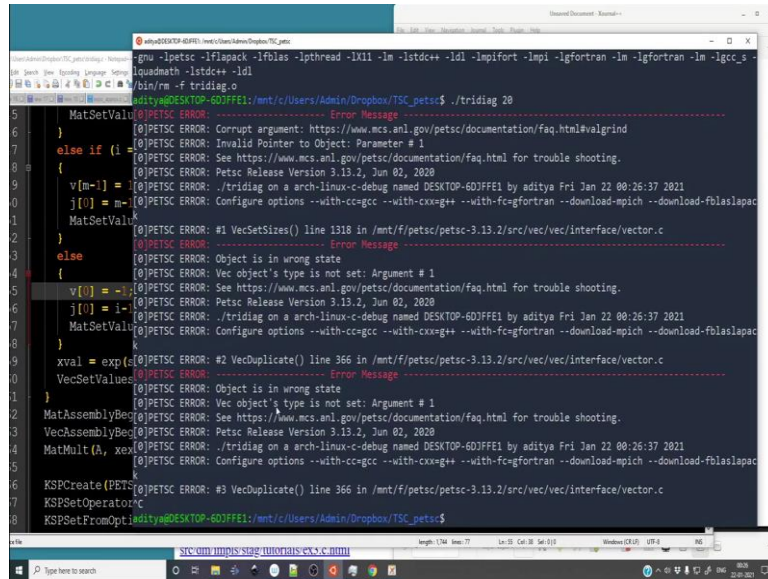
```
4 {
5     v[0] = -1; v[1] = 2; v[2] = -1
6     j[0] = 1-1; j[1] = 1; j[2] = 1+1;
7     MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
8 }
9 xval = exp(sin(i));
10 VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
11 }
12 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
13 VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
14 MatMult(A, xexact, b);
15
16 KSPCreate(PETSC_COMM_WORLD, &ksp);
17 KSPSetOperators(ksp, A, A);
18 KSPSetFromOptions(ksp);
19 KSPSolve(ksp, b, x);
20
21
22 MatDestroy(&A);
23 VecDestroy(&b);
24 VecDestroy(&x); VecDestroy(&xexact);
25 KSPDestroy(&ksp);
26 return PetscFinalize();
27 }
```

So, with the help of this we are going to begin the matrix assembly and we are going to finish it. Similarly, we have to do the vector assembly as well for xexact because we have constructed xexact also inside the loop. So, this will be xexact this will be xexact alright. So, VecAssembly is done.

Now, what we must do is find out what b will be. So, as discussed b will be actually a times x. So, how do we do that? So, we do a MatMult this is an inbuilt function a comma xexact and it will store the solution in b. So, it in it expects an input of the matrix the vector and the output vector. So, we are assigning b A times xexact great.

Now, what we are going to do is we are going to create the KSP object. So, we are going to reuse our code well it helps us save some time. So, ksp A, A and everything else remains the same alright.

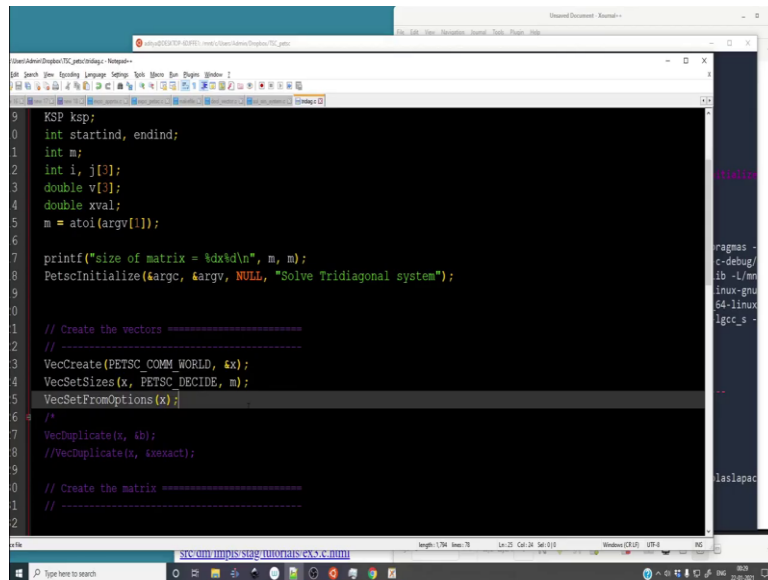
(Refer Slide Time: 49:22)



```
aditya@DESKTOP-60JFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20
MatSetValue [0]PETSC ERROR: Corrupt argument: https://www.mcs.anl.gov/petsc/documentation/faq.html#valgrind
}
else if (i [0]PETSC ERROR: Invalid Pointer to Object: Parameter # 1
[0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
{
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
v[m-1] = [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-60JFFE1 by aditya Fri Jan 22 00:26:37 2021
j[0] = m- [0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc-ffortran --download-mpich --download-flblaslapack
MatSetValue [0]PETSC ERROR: #1 VecSetSizes() line 1318 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/vector.c
[0]PETSC ERROR: Error Message -----
}
else [0]PETSC ERROR: Object is in wrong state
[0]PETSC ERROR: Vec object's type is not set: Argument # 1
v[0] = - [0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
j[0] = 1- [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-60JFFE1 by aditya Fri Jan 22 00:26:37 2021
MatSetValue [0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc-ffortran --download-mpich --download-flblaslapack
xval = exp [0]PETSC ERROR: #2 VecDuplicate() line 366 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/vector.c
VecSetValue [0]PETSC ERROR: Error Message -----
[0]PETSC ERROR: Object is in wrong state
[0]PETSC ERROR: Vec object's type is not set: Argument # 1
MatAssemblyBegin [0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
VecAssemblyBegin [0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
MatMult(A, x, [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-60JFFE1 by aditya Fri Jan 22 00:26:37 2021
[0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc-ffortran --download-mpich --download-flblaslapack
KSPCreate(PETSC [0]PETSC ERROR: #3 VecDuplicate() line 366 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/vector.c
KSPSetOperator [0]PETSC ERROR: Error Message -----
KSPSetFromOpti [0]PETSC ERROR: Error Message -----
aditya@DESKTOP-60JFFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

Well there appears to be errors and I am not sure and where it pointed. Let us see there appears to be an error in VecDuplicate xexact there it is there is the small error that was causing the segmentation for well that is the risk you run if you are copying snippets around

(Refer Slide Time: 50:20)



```
9 KSP ksp;
0 int startind, endind;
1 int m;
2 int i, j[3];
3 double v[3];
4 double xval;
5 m = atoi(argv[1]);
6
7 printf("size of matrix = %dx%d\n", m, m);
8 PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
9
10
11 // Create the vectors -----
12 // -----
13 VecCreate(PETSC_COMM_WORLD, 4x);
14 VecSetSizes(x, PETSC_DECIDE, m);
15 VecSetFromOptions(x);
16 // -----
17 VecDuplicate(x, sb);
18 //VecDuplicate(x, xexact);
19
20 // Create the matrix -----
21 // -----
22
```

(Refer Slide Time: 50:22)

```
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc/tridiag.c$ gcc -c tridiag.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/tridiag.c:7:12: warning: unused variable 'xexact' [Wunused-variable]
   7 |     Vec x, b, xexact;
     |     ~~~~~
     |     warning: unused variable 'xexact' [Wunused-variable]

   9 | KSP ksp;
     |
     | int startind,
     | int m;
     |
     | int i, j[3];
     |
     | double v[3];
     | double xv[3];
     | m = atoi(argv[2]);
     | printf("size
     | PetscInitializ
     | Mat;
     | // Create the v
     | // -----
     | VecCreate(PETS
     | VecSetSizes(x
     | VecSetFromOpt
     | //
     | VecDuplicate(x
     | //VecDuplicate
     | // Create the m
     | // -----
     | /bin/rm -f tridiag.o
     | aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20
     | size of matrix = 20x0
     | // Create the v
     | // -----
     | VecCreate(PETS
     | VecSetSizes(x
     | VecSetFromOpt
     | //
     | VecDuplicate(x
     | //VecDuplicate
     | // Create the m
     | // -----
     | /bin/rm -f tridiag.o
     | aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make tridiag
     | /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o tridiag.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-u
     | // Create the m
     | // -----
     | /arch-linux-c-debug/include      pwd /tridiag.c
```

(Refer Slide Time: 50:25)

```
aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ gcc -c tridiag.c
/mnt/c/Users/Admin/Dropbox/TSC_petsc/tridiag.c:12:6: warning: unused variable 'i' [Wunused-variable]
   12 |     int i, j[3];
     |     ~~~
     |     warning: unused variable 'i' [Wunused-variable]

   9 | KSP ksp;
     |
     | int startind,
     | int m;
     |
     | int i, j[3];
     | double v[3];
     | double xv[3];
     | m = atoi(argv[2]);
     | KSP ksp;
     | printf("size
     | PetscInitializ
     | Mat;
     | // Create the v
     | // -----
     | VecCreate(PETS
     | VecSetSizes(x
     | VecSetFromOpt
     | //
     | VecDuplicate(x
     | //VecDuplicate
     | // Create the m
     | // -----
     | /bin/rm -f tridiag.o
     | aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

(Refer Slide Time: 50:27)

```

1 int j[3];
2
3 KSP ksp;
4 int startind,
5 int m;
6 int i, j[3];
7 double v[3];
8 double xval;
9 m = atoi(argv[1]);
10 Mat A;
11 PetscInitializ
12 Vec x, b, xexact;
13 // Create the v
14 VecCreate(PETS
15 VecSetSizes(x
16 VecSetFromOpti
17 VecDuplicate(x
18 //VecDuplicate
19 loadmath -lstdc+ -ldl
20 /bin/rm -f tridiag.o
21 // Create the m
22 size of matrix = 20x20
23 // -----
24 aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20
25 aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$

```

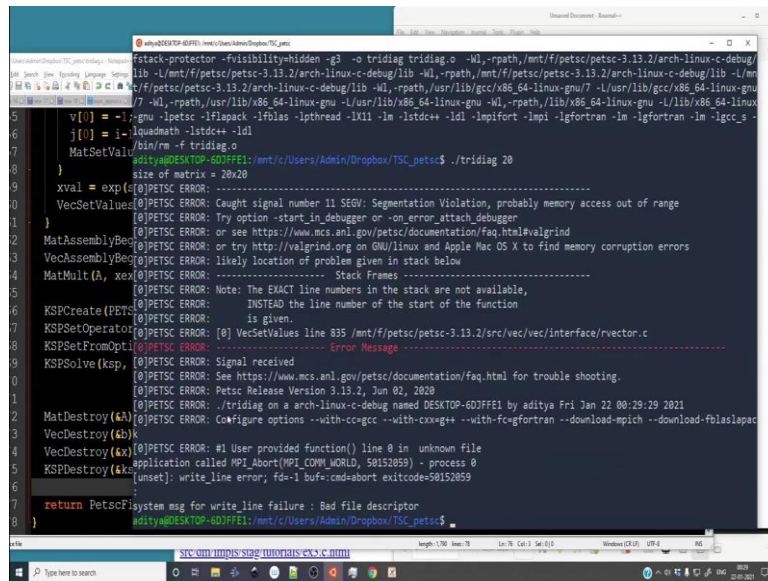
(Refer Slide Time: 50:40)

```

1 Mat A;
2 Vec x, b, xexact;
3 v[0] = -1;
4 j[0] = i-1;
5 MatSetValue
6 xval = exp
7 KSPCreate(PETS
8 KSPSetOperato
9 KSPSetFromOpti
10 KSPSolve(ksp,
11 MatDestroy
12 VecDestroy
13 VecDestroy
14 KSPDestroy
15 return PetscF

```

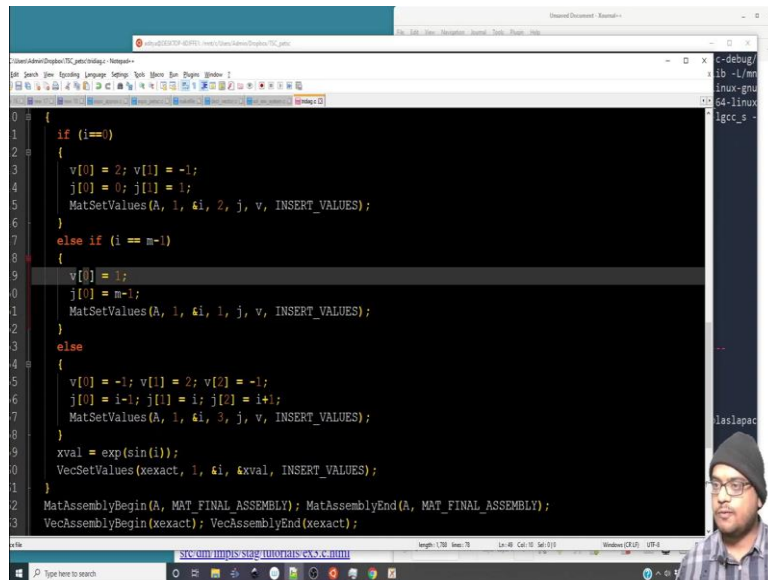
(Refer Slide Time: 50:43)



```
ftack-protector -fvisibility=hidden -g3 -o tridiag tridiag.o -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/...
5 v[0] = -1;
6 j[0] = j-1;
7 MatSetVal...
8 size of matrix = 20x20
9 xval = exp(s[0]);
10 VecSetValues(A, xex[0]);
11 }
12 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
13 VecAssemblyBegin(A, MAT_FINAL_ASSEMBLY);
14 MatMult(A, xex[0]);
15 KSPCreate(Petsc);
16 KSPSetOperator(KSP, A);
17 KSPSetFromOptions(KSP);
18 KSPSolve(KSP, x);
19 MatDestroy(A);
20 VecDestroy(x);
21 KSPDestroy(KSP);
22 return PetscPrintf(stderr, "Bad file descriptor\n");
23 }
```

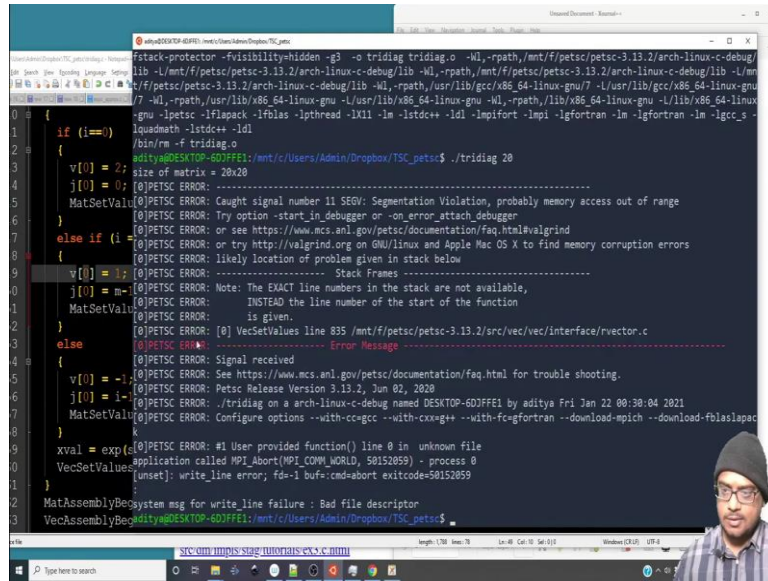
And well over time you learn to be proficient in tracking down the bugs, but the way I have tracked down this bug is not an efficient way of tracking it, but hey if it works it is not stupid well there is some error let us see; what is the error segmentation fault out of range let us see.

(Refer Slide Time: 51:12)



```
0 {
1   if (i==0)
2   {
3     v[0] = 2; v[1] = -1;
4     j[0] = 0; j[1] = 1;
5     MatSetValues(A, 1, &i, 2, j, v, INSERT_VALUES);
6   }
7   else if (i == m-1)
8   {
9     v[0] = 1;
10    j[0] = m-1;
11    MatSetValues(A, 1, &i, 1, j, v, INSERT_VALUES);
12  }
13  else
14  {
15    v[0] = -1; v[1] = 2; v[2] = -1;
16    j[0] = i-1; j[1] = i; j[2] = i+1;
17    MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
18  }
19  xval = exp(sin(i));
20  VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
21 }
22 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
23 VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
24 }
```

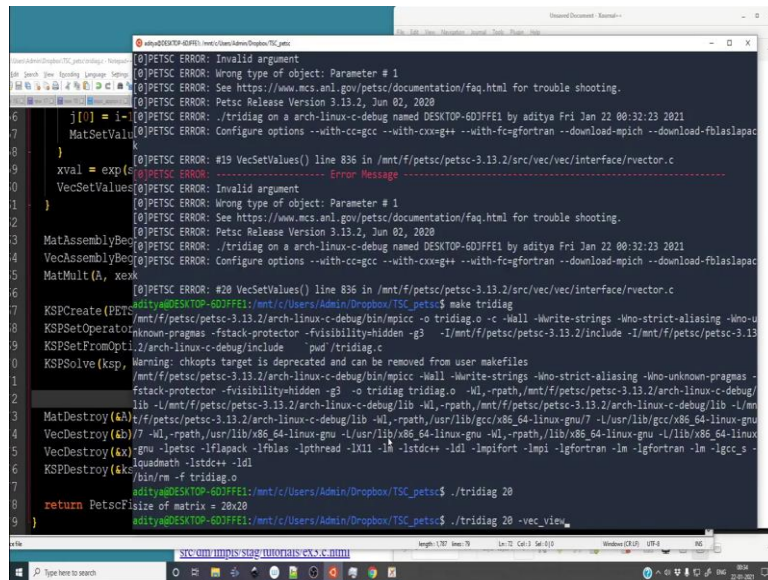
(Refer Slide Time: 51:16)



```
ftack-protector -fvvisibility=hidden -g3 -o tridiag tridiag.o -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,-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,-rpath,/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 -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lflblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgcc_s -lm -ldl -lm -lquadmath -lstdc++ -ldl
/bin/rm -f tridiag.o
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20
size of matrix = 20x20
j[0] = 0; [0]PETSC ERROR: -----
MatSetValue [0]PETSC ERROR: Caught signal number 11 SEGV: Segmentation Violation, probably memory access out of range
[0]PETSC ERROR: Try option -start_in_debugger or -on_error_attach_debugger
[0]PETSC ERROR: or see https://www.mcs.anl.gov/petsc/documentation/faq.html#valgrind
else if (i [0]PETSC ERROR: or try http://valgrind.org on GNU/Linux and Apple Mac OS X to find memory corruption errors
[0]PETSC ERROR: Likely location of problem given in stack below
----- Stack Frames -----
v[0] = m- [0]PETSC ERROR: Note: The EXACT line numbers in the stack are not available,
[0]PETSC ERROR: INSTEAD the line number of the start of the function
MatSetValue [0]PETSC ERROR: is given.
[0]PETSC ERROR: ----- Error Message -----
[0]PETSC ERROR: [0] VecSetValues line 835 /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/rvector.c
[0]PETSC ERROR: Signal received
v[0] = -1; [0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
j[0] = 1- [0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
MatSetValue [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-6D3JFE1 by aditya Fri Jan 22 00:38:04 2021
[0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc=gfortran --download-mpich --download-flblaslapack
xval = exp( [0]PETSC ERROR: #1 User provided function() line 0 in unknown file
[0]PETSC ERROR: application called MPI_Abort(MPI_COMM_WORLD, 50152059) - process 0
VecSetValues [unset]: write_line error: fd=1 buf=0x00000000 exitcode=50152059
MatAssemblyBeo [0]PETSC ERROR: write_line failure: Bad file descriptor
VecAssemblyBeo [0]PETSC ERROR: write_line failure: Bad file descriptor
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

There is an very obvious error well still some error let us try to fix it well that was a very stupid mistake.

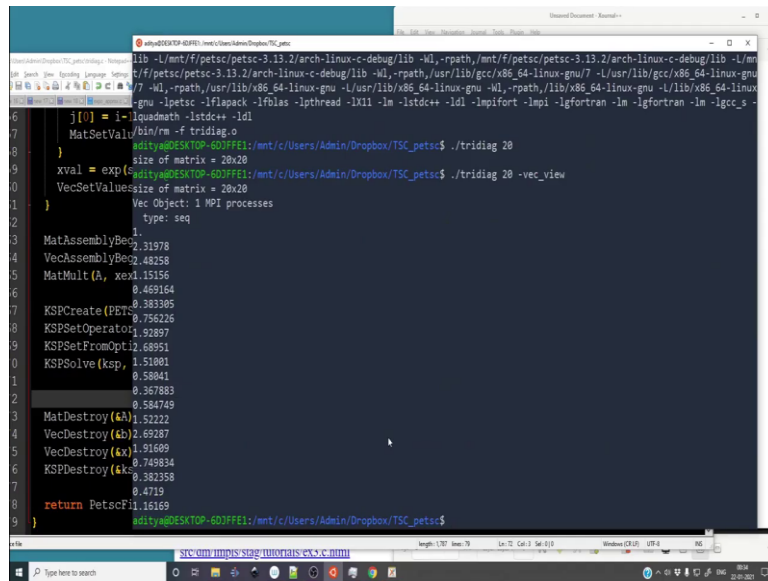
(Refer Slide Time: 51:50)



```
[0]PETSC ERROR: Invalid argument
[0]PETSC ERROR: Wrong type of object: Parameter # 1
[0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
j[0] = 1- [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-6D3JFE1 by aditya Fri Jan 22 00:32:23 2021
MatSetValue [0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc=gfortran --download-mpich --download-flblaslapack
xval = exp( [0]PETSC ERROR: #19 VecSetValues() line 836 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/rvector.c
VecSetValues [0]PETSC ERROR: Invalid argument
[0]PETSC ERROR: Wrong type of object: Parameter # 1
[0]PETSC ERROR: See https://www.mcs.anl.gov/petsc/documentation/faq.html for trouble shooting.
[0]PETSC ERROR: Petsc Release Version 3.13.2, Jun 02, 2020
MatAssemblyBeo [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-6D3JFE1 by aditya Fri Jan 22 00:32:23 2021
VecAssemblyBeo [0]PETSC ERROR: Configure options --with-cc-gcc --with-cxx-g++ --with-fc=gfortran --download-mpich --download-flblaslapack
MatMult(A, xexx
[0]PETSC ERROR: #20 VecSetValues() line 836 in /mnt/f/petsc/petsc-3.13.2/src/vec/vec/interface/rvector.c
KSPCreate(PET [0]PETSC ERROR: ./tridiag on a arch-linux-c-debug named DESKTOP-6D3JFE1 by aditya Fri Jan 22 00:32:23 2021
KSPSetOperator [0]PETSC ERROR: Wrong type of object: Parameter # 1
KSPSetFromOpti [0]PETSC ERROR: Wrong type of object: Parameter # 1
KSPSolve(ksp, 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 -ftack-protector -fvvisibility=hidden -g3 -I/mnt/f/petsc/petsc-3.13.2/include -I/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,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lflblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgcc_s -lm -ldl -lm -lquadmath -lstdc++ -ldl
/bin/rm -f tridiag.o
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20
size of matrix = 20x20
aditya@DESKTOP-6D3JFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20 -vec_view
```

Well this kind of things to happen. So, everything's working nice and well. So, we have been able to solve till KSPSolve so far and now we would like to view the vectors. So, let us do that. So, - vec view ok. So, instead of doing a - vec view let us do vec view inside the code before destroying them we are going to do a vector of x xexact ok. Let us see let us recompile it and ok.

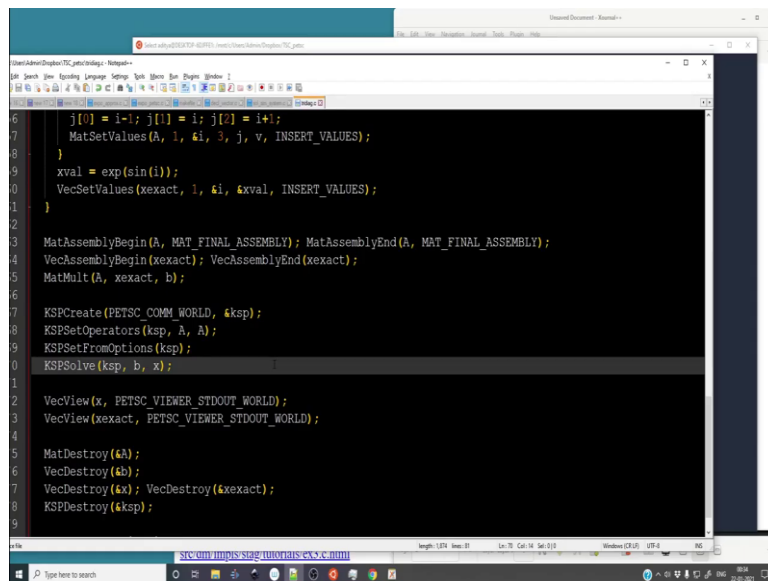
(Refer Slide Time: 52:18)



```
lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/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 -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux-gnu -lpetsc -lflapack -lfbblas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgcc_s -lgnu

6 j[0] = 1;
7 MatSetVal(bin/rm -f tridiag.o
8 size of matrix = 20x20
9 xval = exp
10 VecSetValues(size of matrix = 20x20
11 Vec Object: 1 MPI processes
12 type: seq
13 MatAssemblyBegin
14 VecAssemblyBegin
15 MatMult(A, xval,
16 0.469164
17 KSPCreate(PET
18 KSPSetOperator
19 KSPSetFromOptions
20 KSPSolve(ksp,
21 0.58841
22 0.367883
23 MatDestroy(&A)
24 VecDestroy(&b)
25 VecDestroy(&x)
26 KSPDestroy(&ksp)
27 0.382358
28 0.4719
29 return PetscPi
30 aditya@DESKTOP-6D3FFE1: /mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

(Refer Slide Time: 52:30)



```
6 j[0] = 1-1; j[1] = i; j[2] = i+1;
7 MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
8 }
9 xval = exp(sin(i));
10 VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
11 }
12
13 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14 VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
15 MatMult(A, xexact, b);
16
17 KSPCreate(PETSC_COMM_WORLD, &ksp);
18 KSPSetOperators(ksp, A, A);
19 KSPSetFromOptions(ksp);
20 KSPSolve(ksp, b, x);
21
22 VecView(x, PETSC_VIEWER_STDOUT_WORLD);
23 VecView(xexact, PETSC_VIEWER_STDOUT_WORLD);
24
25 MatDestroy(&A);
26 VecDestroy(&b);
27 VecDestroy(&x); VecDestroy(&xexact);
28 KSPDestroy(&ksp);
```

(Refer Slide Time: 52:36)

```

aditya@DESKTOP-60JFFE1: ~/c/Users/Admin/Dropbox/TSC_petsc
1 0.58841
2 0.367883
3 0.584749
4 1.52222
5 2.31978
6 j[0] = 1 - 0.383395
7 MatSetVal 0.755226
8 }
9 xval = exp(-1.51001)
10 VecSetValues 0.58841
11 }
12 0.367883
13 0.584749
14 1.52222
15 MatAssemblyBe 2.69287
16 VecAssemblyBe 1.91689
17 MatMult(A, xex 0.749834
18 0.382358
19 KSPCreate(PET 0.4719
20 KSPSetOperat 1.16169
21 KSPSetFromOpt /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o tridiag.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-u
22 KSPSolve(ksp, /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/include "pd"/tridiag.o
23 Warning: chkopts target is deprecated and can be removed from user makefiles
24 VecView(x, PE /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -
25 VecView(xexact /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -o tridiag tridiag.o -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/
26 lib -L/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -L/mn
27 MatDestroy (k /mnt/f/petsc/petsc-3.13.2/arch-linux-c-debug/lib -Wl,-rpath,/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu
28 VecDestroy (k /7 -Wl,-rpath,/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-rpath,/lib/x86_64-linux-gnu -L/lib/x86_64-linux
29 VecDestroy (k -gnu -lpetsc -lflapack -lfflas -lpthread -lX11 -lm -lstdc++ -ldl -lmpifort -lmpi -lgfortran -lm -lgcc_s -
30 KSPDestroy (k /quadsmath -lstdc++ -ldl
31 KSPDestroy (k /bin/rm -f tridiag.o
32 aditya@DESKTOP-60JFFE1:~/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 20 -vec_view

```

(Refer Slide Time: 52:42)

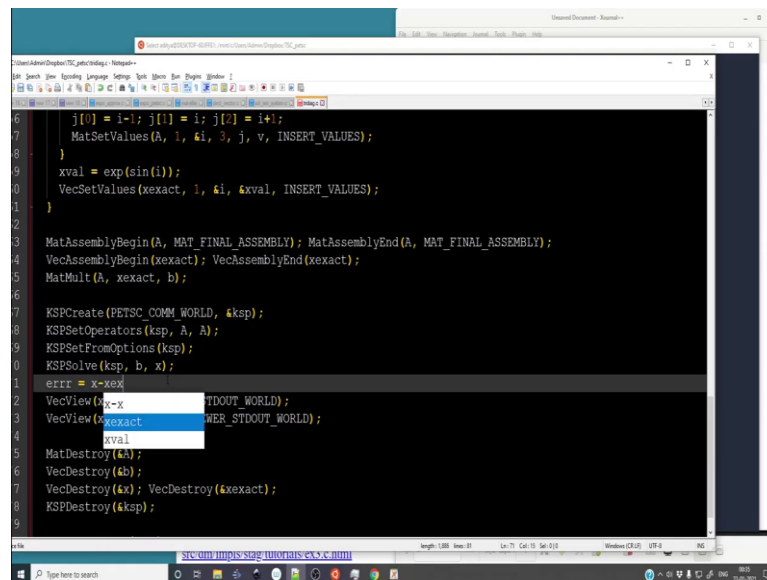
```

aditya@DESKTOP-60JFFE1: ~/c/Users/Admin/Dropbox/TSC_petsc
1 0.58841
2 0.367883
3 0.584749
4 1.52222
5 2.31978
6 j[0] = 1 - 2.69287
7 MatSetVal 1.91689
8 }
9 xval = exp(-0.4719)
10 VecSetValues 1.16169
11 }
12 Vec Object: 1 MPI processes
13 type: seq
14 1.
15 MatAssemblyBe 2.31978
16 VecAssemblyBe 2.48258
17 MatMult(A, xex 1.15156
18 0.469164
19 0.383395
20 KSPCreate(PET 0.755226
21 KSPSetOperat 1.92897
22 KSPSetFromOpt 2.69551
23 KSPSolve(ksp, 1.51001
24 0.58841
25 }
26 VecView(x, PE 0.367883
27 VecView(xexact 0.584749
28 1.52222
29 2.69287
30 1.91689
31 MatDestroy (k / 0.749834
32 VecDestroy (k / 0.382358
33 VecDestroy (k / 0.4719
34 KSPDestroy (k / 1.16169
35 aditya@DESKTOP-60JFFE1:~/c/Users/Admin/Dropbox/TSC_petsc$

```

So, they do look close 1.16169 and well this is quite expected and as a matter of fact what we can do is figure out what the norm will be. So, the norm will be the difference between x and xexact ok. So, in order to find out the norm we must first have an operation x - xexact.

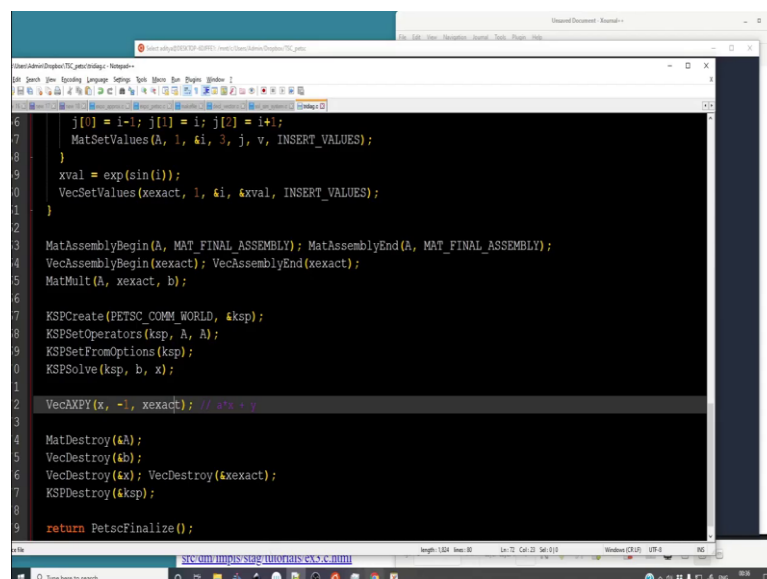
(Refer Slide Time: 53:23)



```
6   j[0] = i-1; j[1] = i; j[2] = i+1;
7   MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
8   }
9   xval = exp(sin(i));
10  VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
11  }
12
13  MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14  VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
15  MatMult(A, xexact, b);
16
17  KSPCreate(PETSC_COMM_WORLD, &ksp);
18  KSPSetOperators(ksp, A, A);
19  KSPSetFromOptions(ksp);
20  KSPSolve(ksp, b, x);
21  errr = x-xex
22  VecView(x, PETSC_STDOUT_WORLD);
23  VecView(xexact, PETSC_STDOUT_WORLD);
24  xval
25  MatDestroy(&a);
26  VecDestroy(&b);
27  VecDestroy(&x); VecDestroy(&xexact);
28  KSPDestroy(&ksp);
29
```

So, we cannot simply do something like what we will do in Python like error equal to $x - x_{exact}$ this is not allowed. So, we must do something which is quite something which resembles something which you would do in blas ok.

(Refer Slide Time: 53:44)

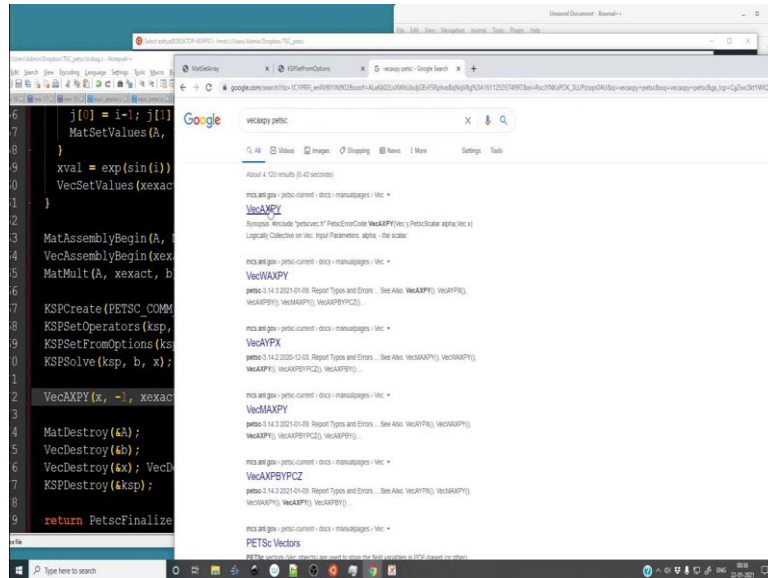


```
6   j[0] = i-1; j[1] = i; j[2] = i+1;
7   MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
8   }
9   xval = exp(sin(i));
10  VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
11  }
12
13  MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
14  VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
15  MatMult(A, xexact, b);
16
17  KSPCreate(PETSC_COMM_WORLD, &ksp);
18  KSPSetOperators(ksp, A, A);
19  KSPSetFromOptions(ksp);
20  KSPSolve(ksp, b, x);
21
22  VecAXPY(x, -1, xexact); // a*x + y
23
24  MatDestroy(&a);
25  VecDestroy(&b);
26  VecDestroy(&x); VecDestroy(&xexact);
27  KSPDestroy(&ksp);
28
29  return PetscFinalize();
```

So, you would do what is called as $VecAX + Y$. So, what this function does is it does $A * X + Y$ where X and Y are two vectors and A is a scalar ok. So, it does $a * x + y$ so, but here a has to be -1 , x and y have to be x and x_{exact} respectively. So, that essentially

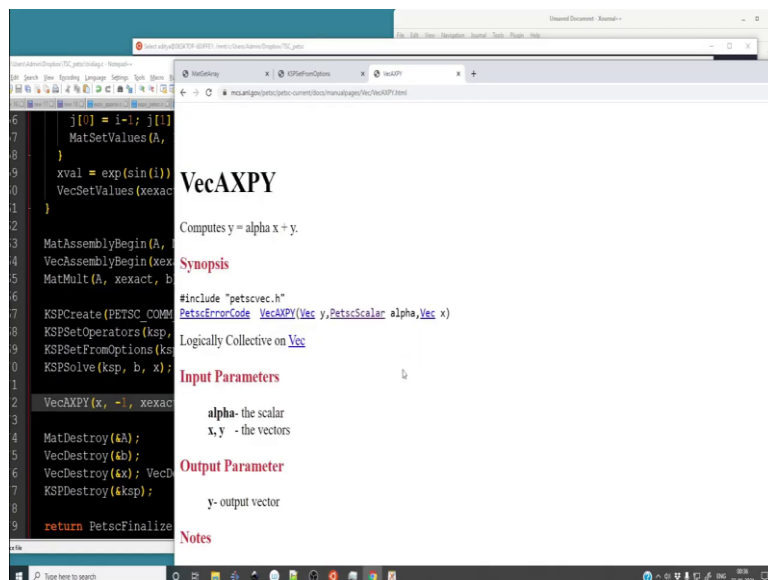
mimics what you want to do and the only caveat is once this operation is done it will overwrite one of the vectors.

(Refer Slide Time: 54:31)



So, let me just show you the function reference and it will tell you exactly what it will overwrite.

(Refer Slide Time: 54:37)



So, it will do $y = \alpha x + y$ ok. So, the first vector gets overwritten.

(Refer Slide Time: 55:00)

```

8 MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
9 }
10 xval = exp(sin(i));
11 VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
12 }
13
14 MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
15 VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
16 MatMult(A, xexact, b);
17
18 KSPCreate(PETSC_COMM_WORLD, &ksp);
19 KSPSetOperators(ksp, A, A);
20 KSPSetFromOptions(ksp);
21 KSPSolve(ksp, b, x);
22
23 VecAXPY(x, -1, xexact); // a*x + y; x -> xexact - x
24 VecNorm(x, NORM_2, &errnorm);
25
26 printf("Grid %d: Error = %f\n", m, errnorm);
27
28 MatDestroy(&b);
29 VecDestroy(&b);
30 VecDestroy(&x); VecDestroy(&xexact);
31 KSPDestroy(&ksp);

```

So, over here x will get overwritten with the x will be overwritten by xexact - x ok. This thing will cause x to be overwritten by xexact - x and that is ok we just wanted to figure out the norm finally, we can do VecNorm that is the norm of the vector x because that contains the error now and the NORM has to be a 1 2 NORM 2 and we will store it inside error of norm. So, error of norm has to be declared as a double.

(Refer Slide Time: 55:39)

```

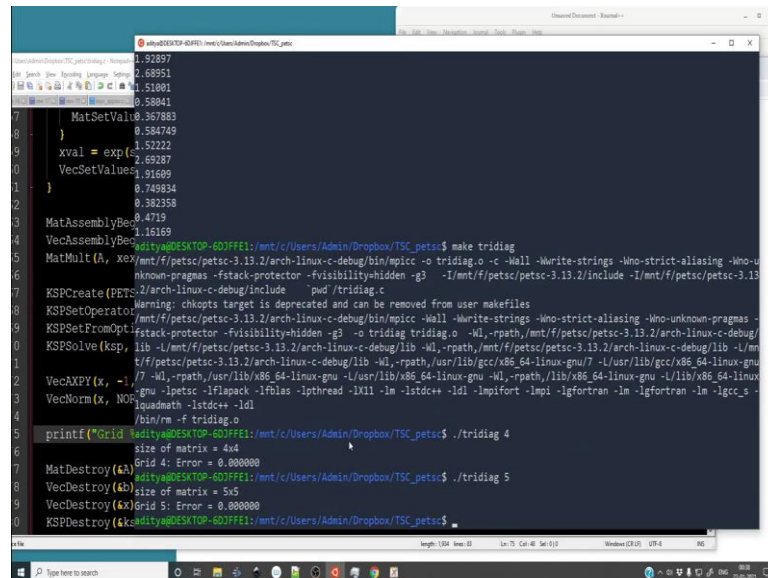
8 Mat A;
9 KSP ksp;
10 int startind, endind;
11 int m;
12 int i, j[3];
13 double v[3];
14 double xval, errnorm;
15 m = atoi(argv[1]);
16
17 printf("Size of matrix = %dx%d\n", m, m);
18 PetscInitialize(&argc, &argv, NULL, "Solve Tridiagonal system");
19
20 // Create the vectors -----
21 // -----
22 VecCreate(PETSC_COMM_WORLD, &x);
23 VecSetSizes(x, PETSC_DECIDE, m);
24 VecSetFromOptions(x);
25
26 VecDuplicate(x, &b);
27 VecDuplicate(x, &xexact);
28
29 // Create the matrix -----
30 // -----

```

So, let us declare it as a double ok. So, we have declared it as a norm. So, finally, before destroying everything we can print out printf Error equal to percentage If errnorm. In fact, we can write down the grid size as well. So, Grid percentage d and error is this we

will print m and error norm. So, it will help us evaluate how the error reduces as the number of grids increases.

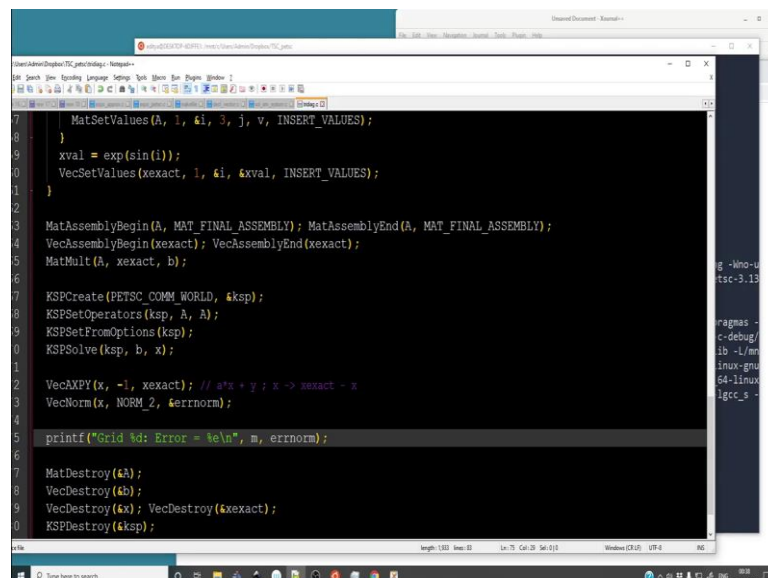
(Refer Slide Time: 56:30)



```
1.92897
2.68951
1.51001
0.58041
MatSetVal 0.367983
0.834749
}
1.52222
xval = exp(2.69287)
VecSetValues 1.91609
0.745834
0.382358
MatAssemblyBe 0.4719
VecAssemblyBe 1.16169
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ make tridiag
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 4
size of matrix = 4x4
Grid 4: Error = 0.000000
MatDestroy (4)
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 5
size of matrix = 5x5
Grid 5: Error = 0.000000
KSPDestroy (6)
aditya@DESKTOP-6D3FFE1:/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

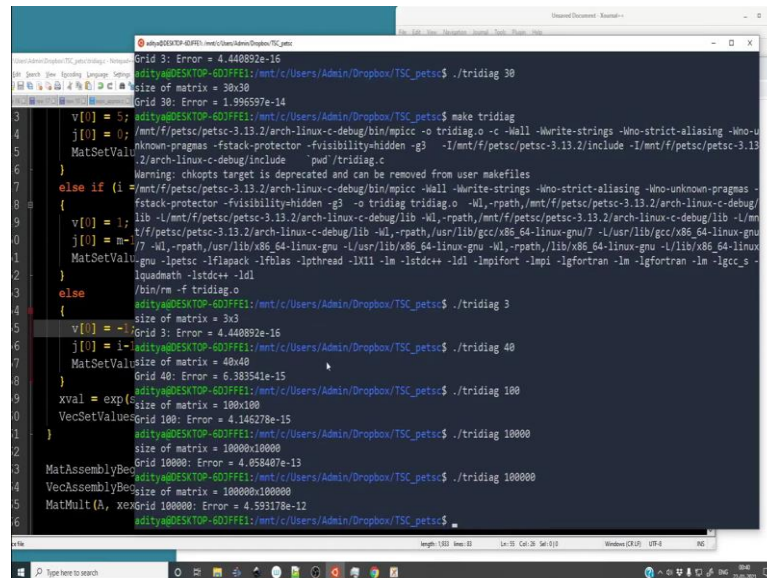
So, let us recompile let us do it with 4 grids, 5 grids well it shows a 0 error let us say percentage e.

(Refer Slide Time: 56:48)



```
MatSetValues(A, 1, &i, 3, j, v, INSERT_VALUES);
}
xval = exp(sin(i));
VecSetValues(xexact, 1, &i, &xval, INSERT_VALUES);
}
MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY);
VecAssemblyBegin(xexact); VecAssemblyEnd(xexact);
MatMult(A, xexact, b);
KSPCreate(PETSC_COMM_WORLD, &ksp);
KSPSetOperators(ksp, A, A);
KSPSetFromOptions(ksp);
KSPSolve(ksp, b, x);
VecAXPY(x, -1, xexact); // a*x + y; x -> xexact - x
VecNorm(x, NORM_2, &ernorm);
printf("Grid %d: Error = %e\n", m, ernorm);
MatDestroy(&A);
VecDestroy(&b);
VecDestroy(&x); VecDestroy(&xexact);
KSPDestroy(&ksp);
```

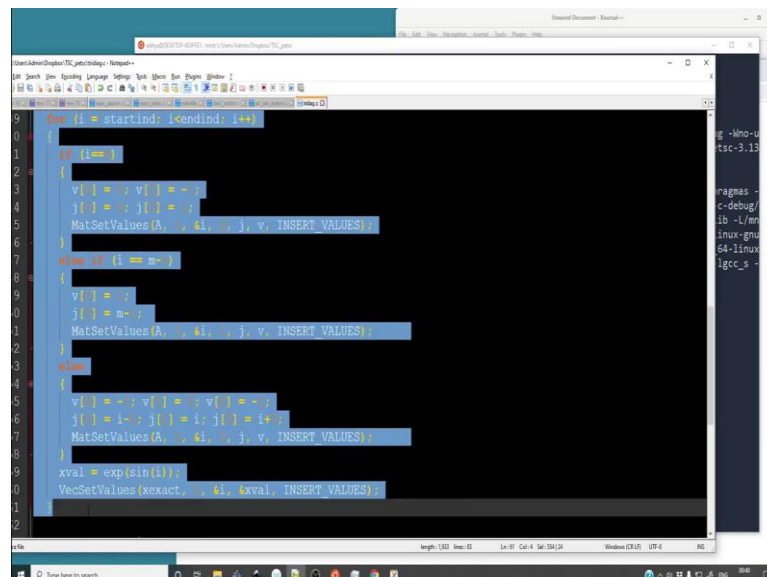

(Refer Slide Time: 57:47)



```
Grid 3: Error = 4.440892e-16
aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 30
size of matrix = 30x30
Grid 30: Error = 1.996597e-14
3 v[] = 5;
4 j[] = 0;
5 MatSetVal...
6 size of matrix = 30x30
7 Warning: chkopts target is deprecated and can be removed from user makefiles
8 else if (i = /mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/bin/mpicc -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unknown-pragmas -fstack-protector -fvvisibility-hidden -g3 -I/mnt/ff/petsc/petsc-3.13.2/include -I/mnt/ff/petsc/petsc-3.13.2/arch-linux-c-debug/include -ped -ftridiag.c
9 {
10 v[] = 1;
11 j[] = m-7;
12 MatSetVal...
13 size of matrix = 3x3
14 else
15 aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 3
16 size of matrix = 3x3
17 Grid 3: Error = 4.440892e-16
18 v[] = -1;
19 j[] = i-1;
20 MatSetVal...
21 size of matrix = 40x40
22 Grid 40: Error = 6.383541e-15
23 aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 100
24 size of matrix = 100x100
25 Grid 100: Error = 4.146278e-15
26 aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$ ./tridiag 10000
27 size of matrix = 10000x10000
28 Grid 10000: Error = 4.858407e-13
29 MatAssemblyBeo...
30 VecAssemblyBeo...
31 size of matrix = 100000x100000
32 MatMult(A, xexGrid 100000: Error = 4.593178e-12
33 aditya@DESKTOP-603FFE1:~/mnt/c/Users/Admin/Dropbox/TSC_petsc$
```

Well, you can make very very large matrices, but anyway yeah. So, you can define the matrix like this we used a very synthetic example to set up the problem after setting up the problem we solved with the problem using the KSP object we found out the norm we printed out the error ok.

(Refer Slide Time: 58:36)



```
3 for (i = startind; i<endind; i++)
4 {
5 v[i] = 5;
6 j[i] = 0;
7 MatSetValues(A, ..., &i, ..., j, v, INSERT_VALUES);
8 }
9 else if (i == m-)
10 {
11 v[i] = 1;
12 j[i] = m-7;
13 MatSetValues(A, ..., &i, ..., j, v, INSERT_VALUES);
14 }
15 }
16 {
17 v[i] = -1; v[j] = 1; v[i+1] = 4;
18 j[i] = i-1; j[i] = i; j[i] = i+1;
19 MatSetValues(A, ..., &i, ..., j, v, INSERT_VALUES);
20 }
21 }
22 xval = exp(sin(i));
23 VecSetValues(xexact, ..., &i, &xval, INSERT_VALUES);
```

So, in what in the next few lectures we will use this kind of program to sort of solve some PDEs. The rest of the code structure will remain more or less the same and I sincerely request you to have a look at this program write your own program. So, that

