

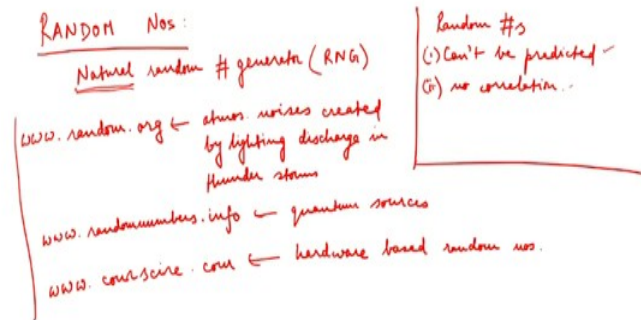
**Computational Physics**  
**Dr. Apratim Chatterji**  
**Dr. Prasenjit Ghosh**  
**Department of Physics**  
**Indian Institute of Science Education and Research, Pune**

**Lecture - 09**  
**Numerical Integration Part 04**

In the next couple of slides, what I am going to do is I am going to tell you briefly about, random numbers, how to test it and so on and so forth, and in Fortran how you can call the random number generator, subroutines to generate random numbers. So, remember I mean, so this will be a very brief recap or of random number generators and random number in general. The whole idea on the concept of random numbers it can be an independent course in itself is such a vast area. Even the testing of random numbers we will talk just about some 3 4 very basic methods of testing, your random numbers which you are using or the random number generator which you are using and that there are also other much more sophisticated methods which we are not going to do.

So, the idea of this part is to make you aware that since you are using a random number which is generated by someone for you, so you should be careful in using that. So, you should be careful and you should test that whether the random number, that the random number generator you are using is indeed random number or there are some correlations in that.

(Refer Slide Time: 01:37)



So, now coming to a slight diversion . So, we will see random numbers. So, by definition a random number means is that it will or random number generator means that it produces a sequence of numbers for which the correlation between any two numbers is 0. So, random numbers can be defined into two categories, one is natural or rather random number generators; natural random number generator which I am calling as RNG. RNG I am using as a short form for a random number generator.

So, what it means is that this type of that random number generator they gives you are genuine random numbers; that means, you cannot predict the random numbers apriori or you cannot find any correlation between them. So, maybe I just write down what are random numbers, the characteristics of random numbers are cannot be predicted. Second, no correlation.

So, random numbers generating series of random numbers which satisfy both these properties in a idle way is typically done by taking the by different ways. So, for example, there are some random number generators which uses the time gap or between two radioactive decays or there are some random number generators which for example, uses the time between the atmospheric noise which is created between to discharge in the atmosphere. So, and there are different websites which can which will give you such numbers. So, I will just list you some of them. So, one is www dot random dot org. So, this is based on atmospheric noises created by lightning discharge in thunderstorms. Then,

there is another website called www dot random numbers dot info. So, they use quantum sources.

So, what I mean by quantum sources is for example. So, we know that a light is comprised of photons. So, what you do is you take light and you take a semi transparent mirror and then you ask the question, will that light be reflected or will that light be transmitted. So, no one can say this apriori, it is a completely random process. So, this is another type of source. Then there is another website which is called www dot c o m s c i r e comscire dot com. So, basically what they use is the noise in the generated by the hardware of our machines. So, basically these are called hardware based random numbers.

So, these are some examples of natural random number. What I mean by that is I mean these are sort of really random. So, you cannot predict what the next number is, you cannot and there is no correlation between them. But apart from these sources, so our computers also has random number generators. So, these are called pseudo random number generator.

(Refer Slide Time: 06:18)

Pseudo-RNG  
 - linear congruence / modular generator  

$$x_i = (a \cdot x_{i-1} + c) \bmod m, i > 0$$

$$x_0 = 2$$

$$x_i = (6x_{i-1} + 7) \bmod 5$$

$$2, 4, 1, 3, 0, 2, 4, 1, 3, 0, \dots$$

$x_i, a, c, m \rightarrow$  all integers  $i=0 \Rightarrow x_i \rightarrow$  called the seed  
 $m \rightarrow$  determines the period  
 $a, c$  are crucial to determine the period

$x_i = (27x_{i-1} + 11) \bmod 54$   
 $x_0 = 0$   
 $11, 38, 11, 38, 11, 38, \dots$

Shift register



So, this random subroutines are present comes along with the type of compiler you are using. For example, in Fortran you have certain subroutines which generates you set of random numbers which are which are which are which are which are which are sort of random to some extent, but it is

not completely random in the true sense the reason for that is these are basically generated by using some mathematical expressions.

So, the most commonly used pseudo random number generator is based on this method called linear congruent or module gen modulo generator. So, basically what it does is it uses equation of this type  $x_i = a \cdot x_{i-1} + c \pmod{m}$  for  $i$  greater than 0. And each of these, so basically  $x_i$ ,  $a$ ,  $c$ ,  $m$  these are all integers. Since, it does a modulo mathematics so that is why this is called modular generator, ok.

And from this expression it is also clear that the value of  $m$  determines the period, but at the same time I would also like to mention that the values of  $a$  and  $c$  are also crucial to determine the period. What I mean is that having a large value of  $m$  with an improper value of  $a$  and  $c$  will not guarantee you our set of numbers which are close which you can call random. So, what I mean is the period of the random number?

So, by the period I mean suppose I generate a random number at  $i$ , then after how many more numbers will I get back the same number or I get back the same series. So, to emphasize the fact that  $a$  and  $c$  choice of  $a$  and  $c$  are also crucial in addition to the choice of  $m$  to get a large series of random number I take you through this following example. Suppose, I choose a case like  $x_i$ , I generate using this  $x_i - 1$  and plus 7 modulo 5. So, if I work out the algebra.

So, what I will and, ok. So, another thing I forgot is for  $i$  equals to 0 the corresponding value of  $x$  that is  $x_0$  is called the seed, to generate the random number. So, if I use  $x_0$  is equal to 2 with this expression, so what I will get is the following set of numbers 2, 4, 1, 3, 0 and after that it will be again repeated 2, 4, 1, 3, 0 and so on and so forth. So, it repeats after every 5 numbers as you have would have expected because my modulo here is 5. So, my period is 5. So, after every 5 it comes in, ok.

Now, let us take another case where I use a following expression  $x_i = 27 \cdot x_{i-1} + 1 \pmod{54}$ , and I start with  $x_0 = 0$  as my seed. So, what I would have expected is that the period for this series random number series would have been 54, but when I try to generate this series, so what I find is the following.

So, I get the first number 11, third second number 38. I would have expected some different number, but to many surprise I get again 11, 38, 11, 38 and so on and so forth.

So, basically although I would have thought that I would have got a period of 54 I am in reality getting a period of 2. So, this sort of highlights that what you what one can the importance of these two quantities the a and the c, these two constants how they can also significantly affect the period of the series.

So, this is a very simple example to show how random numbers are typically generated in a computer, but when you actually use the random number generators in our computer uses. So, there they have much more complex form of this linear congruent method. So, that is typically what they use is that method is called shift register method, where here like I have using I am using just one previous number, there they use a combination of such numbers and generate are much longer periods.

So, with this brief introduction as to how natural random numbers and pseudo random numbers are generated, let us see now how one can test the random numbers, but for that let us look at the properties of the random numbers.

(Refer Slide Time: 12:39)

Properties of random nos. and their tests.  
 Generate  $N$  uniform distribution,  $x_i \in [0,1]$   
 $f(x) = 1$

(1) Average ( $\mu$ ) and Standard deviation ( $\sigma$ )  
 Suppose we generate  $N$  random #s  
 what is the  $k$  moment of the set of  $N$  random #s  
 $\langle x^k \rangle = \frac{1}{N} \sum_{i=1}^N x_i^k f(x_i)$   
 For uniform dist  $f(x_i) = 1 \Rightarrow \langle x^k \rangle = \frac{1}{N} \sum_{i=1}^N x_i^k$   
 As  $N \rightarrow \infty$  replace  $\sum$  by  $\int$   
 $\langle x^k \rangle = \int_0^1 dx x^k = \frac{1}{k+1}$   
 $\mu = \langle x^1 \rangle = \frac{1}{2}$        $\sigma = \sqrt{\langle x^2 \rangle - \langle x \rangle^2} = \frac{1}{\sqrt{12}} = 0.2886$   
 $\mu = 0.5$        $\sigma = 0.2886$



So, what we are going to discuss now is the properties of random numbers and their tests. So, typically when you use the machine dependent random numbers that come from a random number generator that comes along with Fortran, so what they do is they generate random numbers with uniform distribution. So, this is also true. For example, for C compilers also, but the difference between the Fortran and the C compilers are that in the Fortran compiler, they lie between 0 and 1.

So, what one means is by a uniform distribution is that the probability is equal to 1. The probability of finding any number in an interval between 0 to 1 is 1. So, once we know the form of the distribution, so what we can do to test whether it is. So, we know that random numbers generated has this uniform distribution.

So, what we will need to do is first of all to verify that the random numbers which my random number generator giving is it a uniform distribution. So, how does one verify that? To verify that one needs to know the properties of uniform distribution of uniform distribution. By properties I mean, so first of all one needs to know what is the average which I am calling as  $\mu$  and what is the standard deviation which I am calling as  $\sigma$ .

So, suppose we have we generate  $N$  random numbers, so we ask the question what is the  $k$ th moment of the set of  $N$  random numbers. So, average of the  $k$ th moment so, that we can write in this way. So,  $\frac{1}{N}$ , the average of the  $k$ th moment;  $k$ th moment means  $x$  to the power  $k$ . So, the average of the  $k$ th moment that is the average of  $x$  to the power  $k$  is given by  $\frac{1}{N} \sum_{i=1}^N x_i^k$  because  $\frac{1}{N} \sum_{i=1}^N 1$  equals to  $\frac{1}{N}$ . So,  $x_i$  to the power  $k$ , and then the probability distribution for; so, this is for any general probability distribution function. This is the general expression.

Now, for uniform distribution what you have as I told you before is that my  $p(x)$  is equal to 1. So, what this implies is that the moment the average value of the moment  $x^k$ , I can write as  $\frac{1}{N} \sum_{i=1}^N x_i^k$  equals to  $\frac{1}{N} \sum_{i=1}^N x_i^k$ . And if I assume that for large  $N$ ; that means, in the limit  $N$  tends to infinity, so what I can do is I can replace the summation by integration. So, what that implies is. And what will be the range of the integration?

The range of the integration will be between 0 to 1 because my  $x$  is will vary from 0 to 1. So, what I can write is average value of the  $k$ th moment is given by the integral of 0 to 1  $dx$ ,  $x$  to the power  $k$  or in other words it will be  $\frac{1}{k+1}$ . Now, what is the average value of this distribution? So, the average value of this distribution  $\mu$  is nothing, but the first moment  $\int_0^1 x dx$ ,  $x$  to the power 1, so and that is given by half, ok.

In similar way, what is the standard deviation? Standard deviation of any distribution is given by the square root of the variance and the variance is given by the average of the square minus the square of the average. And if we plug this in here so we will get as  $\frac{1}{12}$  which is nothing, but equals to 0.2886.

So, for a given random number generator which claims that it generates a uniform distribution, so the first thing one can do in order to test that generator is that whether indeed that random number generator is giving number set of random numbers with uniform distribution is to compute the average value and the standard deviation of the distribution. And what you should get from that average value should be equal to half and your standard deviation sigma should be 1 by root 12 or 0.2886. So, if you do not get these numbers then you can be sure that the random numbers which you have do not have a uniform distribution. So, this is the first test one needs to do.

(Refer Slide Time: 19:33)


(b) Degree of correlation (auto-correlation function)

$$C_k = \frac{\langle x_{i+k} x_i \rangle - \langle x_i \rangle^2}{\langle x_i^2 \rangle - \langle x_i \rangle^2}, \quad \langle x_{i+k} x_i \rangle = \frac{1}{N-k} \sum_{i=1}^{N-k} x_i x_{i+k}$$

$k=0$   $C_k = 1$

For 0 correlation / uncorrelated #s

Reality  $C_k \rightarrow 0$  / very small



The second test is to test how random are these random numbers or in other words what is the correlation between the random numbers. So, what we do is we do a test called degree of correlation, in other words we compute the following auto-correlation function using the following expression.

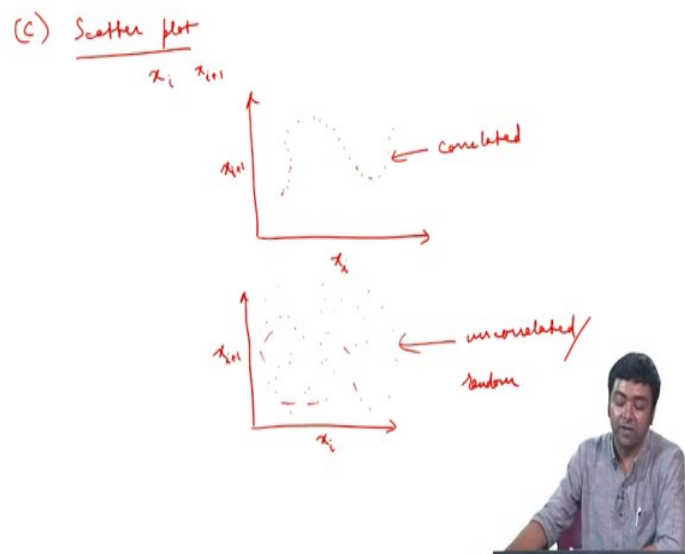
So, we compute this quantity which is calling which we are calling as  $C_k$  the average value of the product of  $i$  plus  $k$  and the  $i$ th random number minus the square of the average value of my distribution and then the average of the square and again the square of the average. So, where this the this term, this first average in the numerator that is average of  $x_i$  plus  $k$ ,  $x_i$  is given by this expression  $\frac{1}{N-k} \sum_{i=1}^{N-k} x_i x_{i+k}$ .

So, now, if we look at this expression carefully, this one and this  $C_k$  expression of  $C_k$  on the right hand side, so if we put  $k$  equals to 0 what you will get is  $C_k$  is equal to 1, because if you put  $k$  equals to 0 this is nothing but the numerator also becomes the variance and denominator we already have the variance, so basically  $C_k$  equals to 1. So, for  $k$  equals to 0  $C_k$  equals to 1 means the correlation of a number with itself with that very same number and that as we expect it to be 1. So, for 0 correlation that is if numbers are completely uncorrelated or on correlated numbers the value of  $C_k$  should be ideally equals to 0, but in reality you never get  $C_k$  to be 0 because there is always some correlation, so  $C_k$  should may be tending towards 0 or very small.

So, if you plot. So, one way to test it is that if you plot your  $C_k$  as a function of  $k$ , so you will find for  $k$  equals to 0 you will find it somewhere here 1,  $C_k$  equals to 1, but after that what we will find is it should very fast, very rapidly decay to 0 and then sort of hover around 0. So, this is what it will look like. So, this is the second test which gives me a measure of how random is my random number.

So, the first test tells me whether my random number distribution, the random numbers which I get from my random number generator have a uniform distribution. The second test this measurement of the degree of correlation gives me how random is the random number. Another way also to test this randomness of random number is by doing something called a scatter plot. So, this is the simplest test.

(Refer Slide Time: 23:12)





So, what is a scatter plot? So, what you do is you take  $x_i$  and  $x_{i+1}$  and then you plot  $x_i$  versus  $x_{i+1}$ . So, for a correlated case you will get something like this. So, you might get something like this. So, this is when things are correlated. But if it is completely uncorrelated then what your scatter plot will look like is something like this it will be spread all over. So, this is my uncorrelated or this will tell how whether it is uncorrelated or random. So, this is visual way of measuring the randomness while the degree of correlation is quantifying the same quantity, ok.

So, now we have seen very briefly how random numbers are generated, we have seen how to test the random numbers which our computer generates it, but now what we will see is how do we get the random numbers in my Fortran program.

(Refer Slide Time: 24:48)

Random # generators in Fortran

`RANDOM_NUMBER(p)`  
 $p \rightarrow$  array/scalar  
 REAL

Fortran gives real RN  
 between [0,1]  
 with uniform distribution

`RANDOM_SEED([SIZE, PUT, GET])`  
 $SIZE \rightarrow$  scalar, integer, INTENT(OUT)  
 No. of integers need to hold the seed  
 $PUT \rightarrow$  Array, integer, INTENT(IN)  
 Size array  $\geq$  # returned by  $SIZE$   
 value to be used as initial seed  
 $GET \rightarrow$  Array, integer, INTENT(OUT)  
 Current value of the seed



So, basically we look at the call to the random number generators in Fortran. So, in Fortran there are two subroutines, and one subroutine is which provides the seed to the random number generator which is called RANDOM underscore SEED and it has 3 arguments, SIZE, PUT and GET.

So, SIZE, this is a scalar quantity, and is an integer, and is an INTENT OUT type of variable or argument. What it means is it always returns an output. So, it gives you that output when you ask. And what information it contains is basically the number of integers used to hold the seed. So, in contrast PUT and GET, both are arrays, both are integers. The difference is put is between, the difference with input and get is put is in

INTENT IN type while get is INTENT OUT type, ok. So, and the size of this array is usually greater than or equal to the number returned by. Same is also true for the GET case also. And what it does is it gives you value to be used as initial seed while so, what you need to put in here is the value to be used to the initial seed while GET will give you the current value of the seed.

So, this is the subroutine related to the seed which is used by this the subroutine which generates the random number and that subroutine is called RANDOM underscore NUMBER, p. So, the argument p it can be either an array or a scalar quantity depending on how you are declaring p in your program and the type is, it is a real argument. So, remember Fortran gives real random numbers lying, real random numbers between 0 and 1 with uniform distribution.

So, now, what we will do is we will see an example as to how these are called in our general program. Also, what we will also see is the importance of this random underscore seed variable, how using this you can control or you can change the series random numbers which you get.

(Refer Slide Time: 30:09)

### Usage of RANDOM NUMBER generator in Fortran

```
PROGRAM ranseed
! how to change the seed for generating random nos.
IMPLICIT NONE

! ----- variables for portable seed setting -----
INTEGER :: l_seed, l, m1
INTEGER, DIMENSION(:), ALLOCATABLE :: a_seed
INTEGER, DIMENSION(10) :: dt_seed
! ----- end of variables for seed setting -----
REAL :: r = 0.0

! ----- Set up random seed portably -----
CALL RANDOM_SEED(size=l_seed)
write(*,*) l_seed
ALLOCATE(a_seed(:l_seed))
CALL RANDOM_SEED(p=a_seed)
CALL DATE_AND_TIME(values=dt_seed)
DO l=1,10
  a_seed(l)=dt_seed(l)
END DO
a_seed(1)=dt_seed(8); a_seed(10)=dt_seed(8)+dt_seed(7)+dt_seed(6)
CALL RANDOM_SEED(p=a_seed)
DEALLOCATE(a_seed)
! ----- Done setting up random seed -----
DO l=1,p
  CALL RANDOM_NUMBER(r)
  WRITE(*,*) "random number is ",r
END DO
END PROGRAM ranseed
```

SIZE → #, INTENT(IN)  
PUT → array; = (IN)  
GET → array; = (OUT)  
DATE\_AND\_TIME  
→ YEAR  
→ MONTH  
→ DAY  
→ DIFF  
→ HR  
→ MIN  
→ SEC  
→ mill sec.



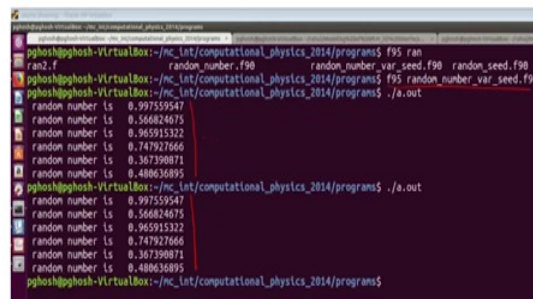
So, here is how our typical program will look like. So, as I mentioned before in the introductory lectures to Fortran, so I start with a program name, then I have the implicit none and then I declare the variables.

So, I am declaring in two types of variables here the INTERGER and a REAL number r. So, in INTEGER, I have 3 variables which are scalar i underscore seed, i is this is just a counter and then n equals to 6, this is also a counter, I mean total number of times i will go. Then I have two INTEGER arrays, one which is an ALLOCATABLE type a underscore seed and another which is a which I have already pre-allocated an area of dimension 1 to 8 which is called dt underscore seed. And then I have real number, I have a real variable r. So, this r is you gives me my random number. So, this part of the program tells you how to initialize the seed.

So, we will come to this shortly, but for the time being we skip this part and what we do is we look into this last part where I am calling my random number. So, what I am trying to do here is I am trying to generate 6 random numbers. So, because I am the reason I am saying 6 is because I have set n equals to 6 and I have a do loop where the counter i goes from 1 to n. I am calling my subroutine random underscore number with the argument r which we have set to a real scalar variable, so and then each time the subroutine is called it gives me a random number which is between 0 and 1 which is stored in this r and I am printing it out and then I am ending the program.

(Refer Slide Time: 32:21)

### Usage of RANDOM NUMBER generator in Fortran



```
pgshosh@pgshosh-VirtualBox:~/nc_int/computational_physics_2014/programs$ f95 ran
ran2.f random_number.f90 random_number_var_seed.f90 random_seed.f90
pgshosh@pgshosh-VirtualBox:~/nc_int/computational_physics_2014/programs$ f95 random_number_var_seed.f90
pgshosh@pgshosh-VirtualBox:~/nc_int/computational_physics_2014/programs$ ./a.out
random number is 0.997559547
random number is 0.566824675
random number is 0.965915322
random number is 0.747927666
random number is 0.387398871
random number is 0.488636895
pgshosh@pgshosh-VirtualBox:~/nc_int/computational_physics_2014/programs$ ./a.out
random number is 0.997559547
random number is 0.566824675
random number is 0.965915322
random number is 0.747927666
random number is 0.387398871
random number is 0.488636895
pgshosh@pgshosh-VirtualBox:~/nc_int/computational_physics_2014/programs$
```



So, if I comment out or if I do not use this first part of the program and if I run it; so, the first time. So, here you see I am compiling the program, but in this program I have what I have done is I have commented out this section set up random seed portability. So,

basically from here the program comes back to here, comes directly to here and it is executed. So, the first time I run, I get this set of 6 random numbers and then if I run it again, one might have thought that I could have got a different set of random numbers, but what I see here is I get again the same 6 number random numbers which is generated here. So, if I run it 100 times, I will get 100 such the same 6 number random numbers which I have got in my first instance of running this program.

So, what this tells is that, I mean since I am not calling the seed here or I am not changing the seed which is being used to generate the set of random numbers, so every time for the same seed, I get the same set of random numbers. So, then the question is how do I change the seed and this is precisely the way (Refer Time: 33:37), I mean this is one way in which n can be done.

So, what I need to do to change the seed is first of all I need to call the subroutine `RANDOM underscore SEED` and then I need to know what is there what is the present seed. And as I mentioned before that the subroutine `underscore seed`, it has two arguments. So, that is it has two arguments, one is `SIZE`, `PUT`, and `GET`. This is a number, all these are integers, this is a number and an `INTENT OUT`. This is an array and again `INTENT OUT`. This is also an array sorry this is `INTENT`, oops; so, this is `INTENT IN`, this is an array which is `INTENT OUT`.

So, first what I do is I inquire from the random number using this size declaration which what is the size of the array which is basically storing the present set of numbers which are used as integer numbers which are used as seed. So, when I and that the output coming from this size is stored in this variable call `i underscore seed`, so which I am printing out. And then what I do is I allocate. So, I as I mentioned before I have this `ALLOCATABLE seed`, a `underscore seed`, so which were I allocate this array, the same dimension as the value stored in this variable `i underscore seed`.

So, once I have done this allocated, then I again call my `RANDOM underscore SEED` and inquire what is the present, what are the present numbers or the array of numbers which gives me the present seed that I get from the `get` command, from this `get` argument here which I store in that array a `underscore seed`.

So, this array now has `i seed` integer numbers. Now, what I need to do is I need to change or reset the value values stored in this a `underscore seed` and send it back again to this

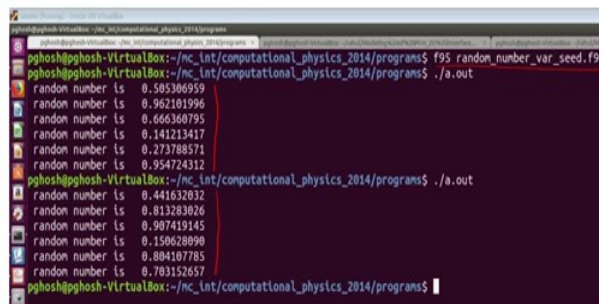
RANDOM underscore SEED, so that this new set of values can be used as a new seed for generating the random number when I run this program a second time for example.

So, how I do it? To do that one way to do that is I called this subroutine called DATE underscore and TIME, and it returns me 8 values which I stored in this variable date underscore seed. So, what are the bad things which this returns me? This DATE, this subroutine DATE underscore and TIME, it gives me the following things it gives me YEAR, MONTH, DAY, then difference between the normal the standard time and the local time, then it gives me time in hours, minutes, seconds and sub seconds, if I remember correctly its milliseconds.

So, these are the 8 things which are given and these 8 numbers I am storing in this variable dt underscore seed. And what I am doing is I am using these numbers stored in this dt underscore seed to reassign some of the values stored under in this array a underscore seed. Once I have done, now I am ready to get with a new set of integer numbers which I can feed to this RANDOM underscore SEED as a new variable and this is precisely what I am doing in this line. So, I am again calling my RANDOM underscore SEED and then using this put equals to a seed I am sending it back, and then the rest part we saw as it is before.

(Refer Slide Time: 38:15)

### Usage of RANDOM NUMBER generator in Fortran



```
gghosh@gghosh-VirtualBox: ~/nc_int/computational_physics_2014/programs$ f95 random_number_var_seed.f90
gghosh@gghosh-VirtualBox: ~/nc_int/computational_physics_2014/programs$ ./a.out
random number is 0.585386959
random number is 0.962181996
random number is 0.666368795
random number is 0.141213417
random number is 0.273788571
random number is 0.954724312
gghosh@gghosh-VirtualBox: ~/nc_int/computational_physics_2014/programs$ ./a.out
random number is 0.441632832
random number is 0.813283826
random number is 0.987419145
random number is 0.158628898
random number is 0.884187785
random number is 0.783152657
gghosh@gghosh-VirtualBox: ~/nc_int/computational_physics_2014/programs$
```



So, now if I run the program, so what I am doing here is I am compiling the program, now in this part I have I am calling my random underscore seed subroutine. So, if I run it

the first time you see I get a set of random numbers. Now, when I run it a second time now the seed I get a new set of random numbers, all the 6 are different from, this then the reason for that is, so when I am running the second time the system clock has changed and as a result of that what is happening is I have a new seed and as a result of that a new set of random numbers are generated. So, this is an example to show how one can generate a new set of random numbers by changing the seed and how one can call the random number generator in the Fortran.

And another thing I would like to mention is that apart from the default random number generator which is already present in your machine in your Fortran compiler, there are also different random number generators some examples you can find for example, in the numerical recipes in C. So, there are some subroutines called ran0, ran1 and so on and so forth.