

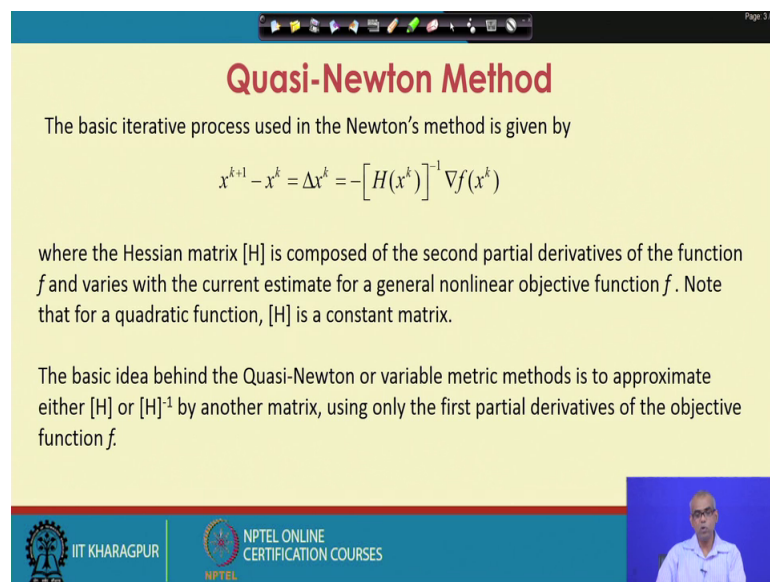
Optimization in Chemical Engineering
Prof. Debasis Sarkar
Department of Chemical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 35

Unconstrained Multivariable Optimization: Gradient Based Methods (Contd.)

Welcome to lecture 35. So, this is the last lecture of week 7 and in this week we have been talking about gradient based methods for optimization of unconstrained multivariable functions. So, in today's lecture we will briefly talk about Quasi-Newton method and Trust Region method and then we will introduce 2 MATLAB functions from MATLAB's optimization toolbox for solutions of unconstrained optimization problems.

(Refer Slide Time: 01:03)



Quasi-Newton Method



The basic iterative process used in the Newton's method is given by

$$x^{k+1} - x^k = \Delta x^k = -[H(x^k)]^{-1} \nabla f(x^k)$$

where the Hessian matrix [H] is composed of the second partial derivatives of the function f and varies with the current estimate for a general nonlinear objective function f . Note that for a quadratic function, [H] is a constant matrix.

The basic idea behind the Quasi-Newton or variable metric methods is to approximate either [H] or [H]⁻¹ by another matrix, using only the first partial derivatives of the objective function f .

Page 3/18

  NPTEL ONLINE CERTIFICATION COURSES

So, let us first briefly talk about Quasi-Newton method. The basic iterative process used in the Newton's method is given by $x^{k+1} - x^k = \Delta x^k = -[H(x^k)]^{-1} \nabla f(x^k)$. We know that the hessian matrix is composed of the second partial derivatives of the function f and the hessian matrix in general will vary with the current estimate for a general non-linear objective function.

Note that for a quadratic function, the hessian is the constant matrix. But for a general non-linear objective function, the hessian matrix will vary with the current estimate of the optimal point. The basic idea behind the Quasi-Newton method or variable metric

methods is to approximate either the matrix H or H inverse that is a hessian or hessian inverse by another matrix using only the first partial derivatives of the objective function.

So, instead of using second partial derivatives of objective function f, we want to replace the hessian matrix or its inverse by another matrix using only first partial derivatives of the objective function f.

(Refer Slide Time: 03:03)

Quasi-Newton Method

If $[H]^{-1}$ is approximated by $[B]$, we can write $x^{k+1} - x^k = \Delta x^k = -[H(x^k)]^{-1} \nabla f(x^k)$ as:

$$x^{k+1} - x^k = \Delta x^k = -\alpha^k [B^k] \nabla f(x^k)$$

Thus the search direction s^k is: $s^k = -[B^k] \nabla f(x^k)$

Note that Steepest Descent method is a special case of above equation with $[B] = [I]$.

Two popular methods:

- Davidon-Fletcher-Powell (DFP) method
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If the inverse of hessian matrix is approximated by a matrix B, we can write as $x^{k+1} - x^k = \alpha^k B^k \nabla f(x^k)$. The search direction then is $s^k = -B^k \nabla f(x^k)$. Now compare this with the Steepest Descent method if B matrix is equal to identity matrix the search direction is $s^k = -\nabla f(x^k)$, which is the search direction for Steepest Descent method.

So, Steepest Descent method becomes a special case of this equation with matrix B equal to identity matrix. There are 2 popular methods Davidon-Fletcher-Powell or DFP method and Broyden-Fletcher-Goldfarb and Shanno or BFGS method.

(Refer Slide Time: 04:40)

Quasi Newton Method

These methods approximate the Hessian matrix using only the first partial derivatives of $f(\mathbf{x})$. Define:

$$\Delta \mathbf{g}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$$

$$\hat{\mathbf{H}}(\mathbf{x}^k) = \hat{\mathbf{H}}^k \quad (\text{approximation of } \mathbf{H}(\mathbf{x}^k))$$

$$\Delta \hat{\mathbf{H}}^k = \hat{\mathbf{H}}^{k+1} - \hat{\mathbf{H}}^k$$

Basically, the methods update an estimate of the Hessian matrix, or it's inverse. Two particular updating mechanisms are:

- >Davidon-Fletcher-Powell (DFP) method
- >Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

These 2 methods update an estimate of the hessian matrix or its inverse. Let us introduce this definitions.

(Refer Slide Time: 05:22)

Quasi Newton Method

Two Updating Mechanisms: (Ref: Edgar and Himmelblau)

Davidon-Fletcher-Powell (DFP) Method:

$$[\Delta \hat{\mathbf{H}}^k]^{-1} = \frac{\Delta \mathbf{x}^k (\Delta \mathbf{x}^k)^T}{(\Delta \mathbf{x}^k)^T \Delta \mathbf{g}^k} - \frac{[\hat{\mathbf{H}}^k]^{-1} \Delta \mathbf{g}^k (\Delta \mathbf{g}^k)^T [\hat{\mathbf{H}}^k]^{-1}}{(\Delta \mathbf{g}^k)^T [\hat{\mathbf{H}}^k]^{-1} \Delta \mathbf{g}^k}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method:

$$\Delta \hat{\mathbf{H}}^k = \frac{\Delta \mathbf{g}^k (\Delta \mathbf{g}^k)^T}{(\Delta \mathbf{g}^k)^T \Delta \mathbf{x}^k} - \frac{\hat{\mathbf{H}}^k \Delta \mathbf{x}^k (\Delta \mathbf{x}^k)^T \hat{\mathbf{H}}^k}{(\Delta \mathbf{x}^k)^T \hat{\mathbf{H}}^k \Delta \mathbf{x}^k}$$

NOTE: In both DFP and BFGS methods, the Hessian matrix method updates remain positive definite. have this property.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

With these definitions in mind, the DFP method or the BFGS methods updates the hessian or hessian inverse as shown by this expression. It may be noted that both DFP and BFGS methods keep the hessian matrix positive definite.

(Refer Slide Time: 06:03)

Trust Region Method

All gradient based methods use: $x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)}$

1. Choose a search direction s^k that will produce descent
2. Perform a line search along this direction

Here emphasis is on finding the descent direction

Trust region methods use the following strategy:

1. Form a “trustworthy” or “safe” approximation of the function $f(x)$ in a “trust region” (neighborhood of the current estimate) in which the current approximation of $f(x)$ produces descent
2. Find the minimum of the approximation

Here emphasis is on finding a trustworthy approximation to $f(x)$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

7

Now, let us briefly define what do you mean by trust region method. We have seen that all gradient based methods use the general scheme x at k plus 1 iteration equal to x at k th iteration plus some step length parameter α^k into search direction at k . So, in case of gradient based method, we choose a search direction s^k that will produce descent. Then perform a line search along this direction s^k .

So, here the emphasis is on finding the descent direction. The trust region methods use a slightly different strategy. In case of trust region methods, we form a trustworthy or safe approximation of the objective function f in a trust region, which is basically the neighborhood of the current estimate in which the current approximation of $f(x)$ produces descent. So, once we have the trustworthy or safe approximation of the function $f(x)$, we find the minimum of this approximate function.

So, here the emphasis is on finding a trustworthy approximation to $f(x)$. So, emphasis shifts from finding the descent direction to finding a trustworthy or safe approximation to function $f(x)$.

(Refer Slide Time: 08:05)

Trust Region Method

Newton or Quasi Newton Method: $\min m(x^k + s) = f(x^k) + \nabla f(x^k)s + \frac{1}{2}s^T H^k s$

Trust Region Method: The model $m(x^k + s)$ is an approximation of $f(x^k + s)$: $m(x^k + s) = f(x^k + s)$
Alternate problem formulation may be:

$$\min m(x^k + s) = f(x^k) + \nabla f(x^k)s + \frac{1}{2}s^T H^k s$$

subject to $\|s\| \leq \delta_k$

Here δ_k is the trust region of the model $m(x)$, i.e. the region where we trust the model is valid.

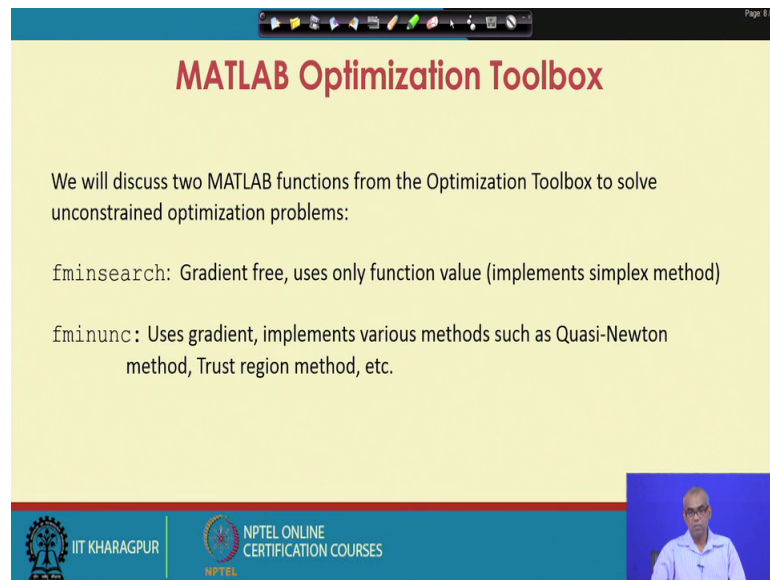
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in case of Newton or Quasi-Newton methods, we solve a problem like this m of x^k plus S equal to f at x^k plus gradient of f at x^k into s plus half S transpose $H^k s$. So, basically compare it with trust series expansion and written only second order terms.

In case of trust region method, the model this $m(x^k + s)$ is basically an approximation of f at $x^k + s$. So, an alternate problem formulation may be we minimize m at $x^k + s$ which is given as f of x^k plus gradient of f at x^k plus into s plus half s transpose $H^k s$ a subject to s confined in a small region. So, norm of s is less or equal to a parameter δ_k . So, δ_k is the trust region of the model m that is the region where the trust where the model is trusted to be valid.

So, basically in case of Newton or Quasi-Newton method the focus is on finding the descent direction, the descent direction s , in case of trust region method we want to find out the approximation of the function in a trust region and minimize the function in the trust region.

(Refer Slide Time: 10:38)

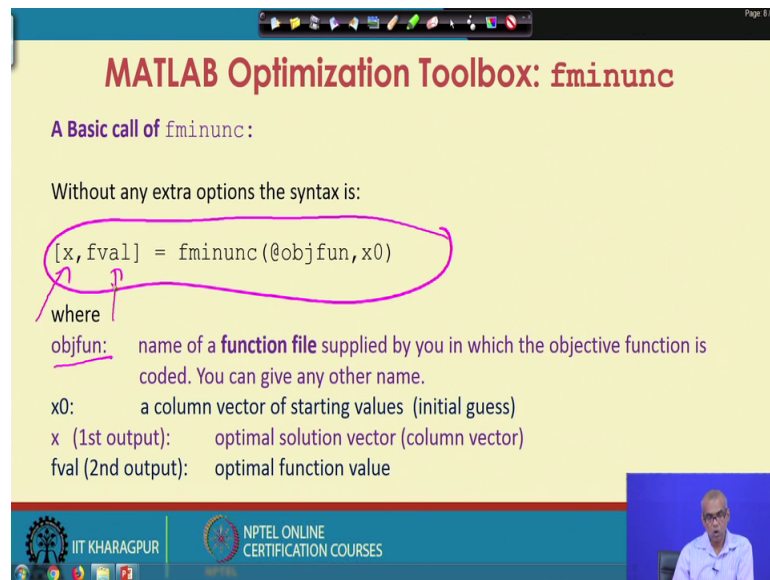


The slide is titled "MATLAB Optimization Toolbox" in red text. Below the title, it states: "We will discuss two MATLAB functions from the Optimization Toolbox to solve unconstrained optimization problems:". It then lists two functions: "fminsearch: Gradient free, uses only function value (implements simplex method)" and "fminunc: Uses gradient, implements various methods such as Quasi-Newton method, Trust region method, etc.". The slide is part of a presentation from IIT Kharagpur and NPTEL Online Certification Courses. A small video inset of a speaker is visible in the bottom right corner.

Now, we will discuss 2 MATLAB functions from the MATLAB optimization toolbox to solve unconstrained optimization problems. In particular we will discuss `fminsearch` and `fminunc`. `fminsearch` is the gradient free method whereas, `fminunc` uses gradient, `fminsearch` being a gradient free method uses only function value method, function value of the objective function.

It is a direct search method, it implements simplex method whereas, `fminunc` uses gradient. So, you have to supply the gradient of the objective function and if we do not supply, it may use numerical computation of the gradient. `fminunc` you implements various algorithms such as Quasi-Newton method trust region method etcetera.

(Refer Slide Time: 12:09)



The slide is titled "MATLAB Optimization Toolbox: fminunc". It contains the following text:

A Basic call of fminunc:

Without any extra options the syntax is:

```
[x, fval] = fminunc(@objfun, x0)
```

where

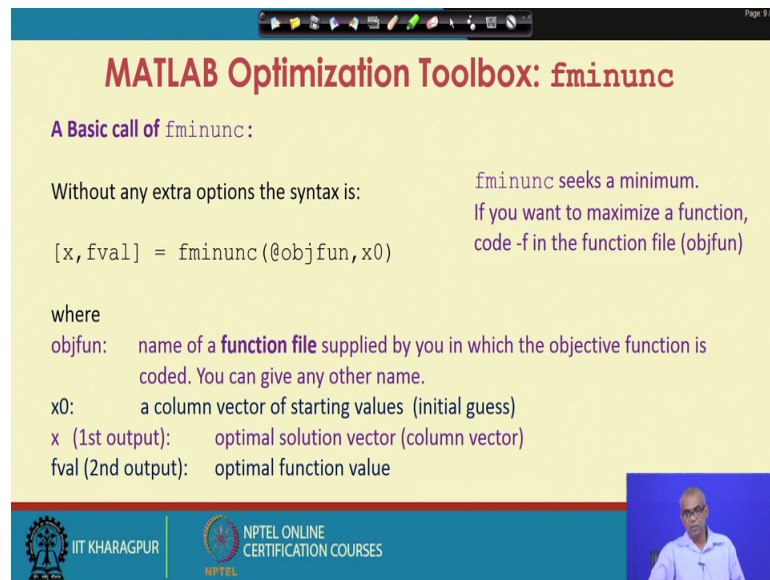
- objfun:** name of a **function file** supplied by you in which the objective function is coded. You can give any other name.
- x0:** a column vector of starting values (initial guess)
- x (1st output):** optimal solution vector (column vector)
- fval (2nd output):** optimal function value

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, and a small video inset of a speaker in the bottom right corner.

So, let us first talk about fminunc, fminunc. A basic call is x comma fval is the function name fminunc at objfun objective function, x0 here objfun is the name of a function file supplied by you in which the objective function is coded in MATLAB language.

It is not necessary that you have to give only this name you are free to supply any other valid name. x 0 is the column vector of starting values; that means, initial guess, the first output x presents the optimal solution vector. Again it is a column vector and the second output fval presents the optimal function value that is the function value at optimally determined point x.

(Refer Slide Time: 14:07)



MATLAB Optimization Toolbox: fminunc

A Basic call of fminunc:

Without any extra options the syntax is: `[x, fval] = fminunc(@objfun, x0)`

*fminunc seeks a minimum.
If you want to maximize a function,
code -f in the function file (objfun)*

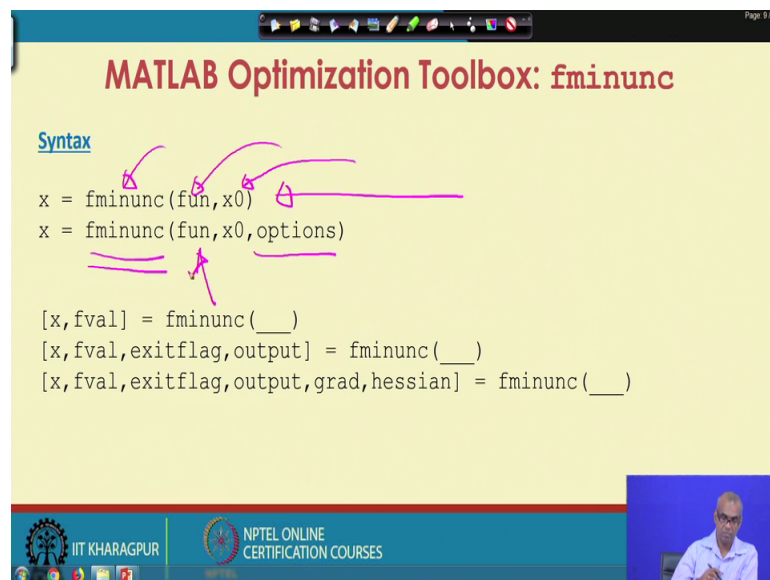
where

- objfun: name of a **function file** supplied by you in which the objective function is coded. You can give any other name.
- x0: a column vector of starting values (initial guess)
- x (1st output): optimal solution vector (column vector)
- fval (2nd output): optimal function value

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

By default fminunc seeks a minimum, if you want to maximize a function you simply code minus f in the function file objfun.

(Refer Slide Time: 14:26)



MATLAB Optimization Toolbox: fminunc

Syntax

`x = fminunc(fun, x0)`
`x = fminunc(fun, x0, options)`

`[x, fval] = fminunc(__)`
`[x, fval, exitflag, output] = fminunc(__)`
`[x, fval, exitflag, output, grad, hessian] = fminunc(__)`

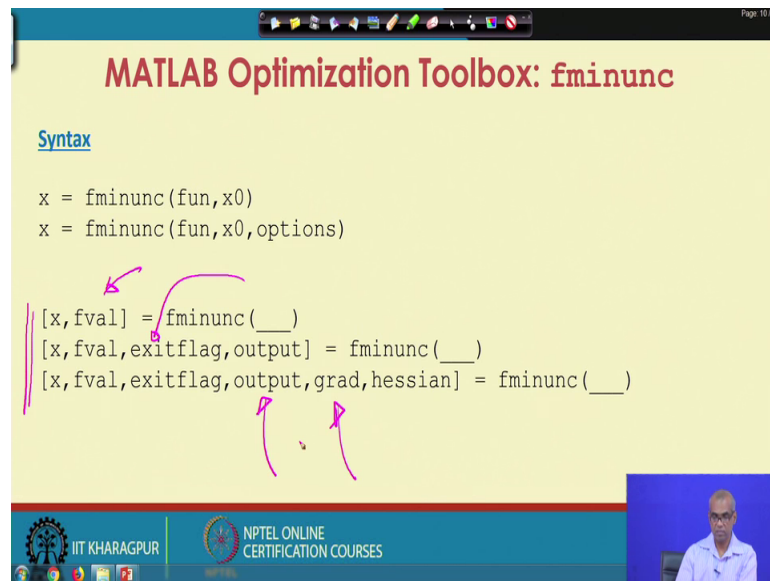
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, these presents some more details about the syntax or how you call the MATLAB function, fminunc note x equal to the function file name, the built in function file name the solver fmin unconstrained fminunc it takes 2 arguments the name of the function file in which you have written, the function that you are going to minimize and the initial

guess. Note this if I supply this if I write this, I have no way of telling MATLAB that I am supplying the gradient which needs to be used.

So, for that you have to add one more argument options and through options you can tell MATLAB or this solver fminunc that you are supplying the gradient of the objective function which can be used for minimization and then this gradient can be written in the same function file where your coding the objective function that you are minimizing.

(Refer Slide Time: 16:20)



The slide is titled "MATLAB Optimization Toolbox: fminunc". It contains the following text:

Syntax

```
x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
```

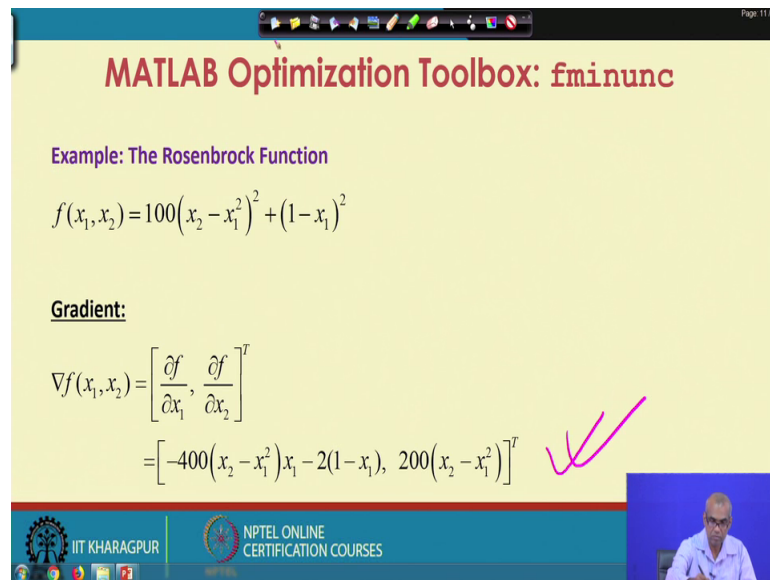
Below the syntax, three lines of code are shown with pink arrows pointing to the output arguments:

```
[x,fval] = fminunc(____)
[x,fval,exitflag,output] = fminunc(____)
[x,fval,exitflag,output,grad,hessian] = fminunc(____)
```

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, and a small video inset of a presenter in the bottom right corner.

Now, here are more details about the syntax if we look at the left hand part see these are the values that are returned by fminunc. So, x is the optimal function value, fval is the function value at optimum x. So, x is the optimal solution and fval is the function value at the optimal solution. Exit flag is a parameter which tells you about whether the optimization was successful how it existed, similarly output is a structure where you have more information's like number of functions value, number of function calls, number of iterations so on and so forth. You can ask the solver to return information on gradient and hessian as well.

(Refer Slide Time: 17:39)



MATLAB Optimization Toolbox: fminunc

Example: The Rosenbrock Function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

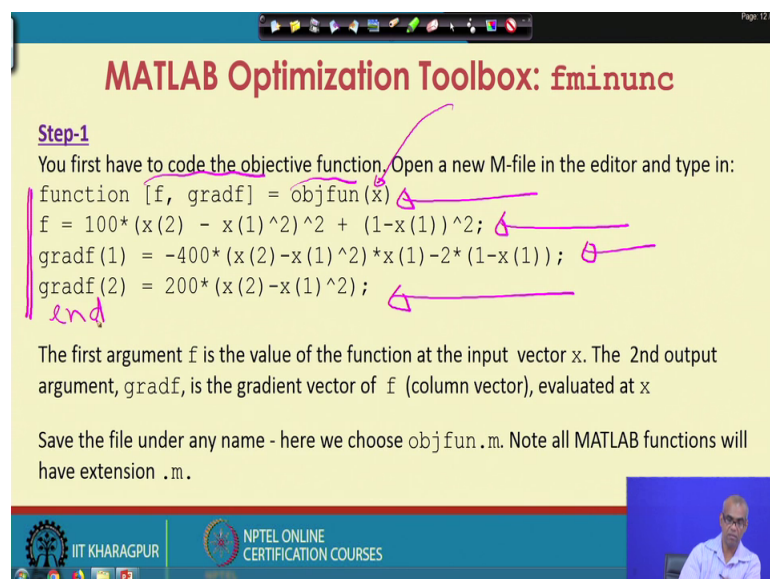
Gradient:

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}^T$$
$$= \begin{bmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}^T$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, now let us take an example, let us consider the Rosenbrock function which is the 2 dimensional function $100(x_2 - x_1^2)^2 + (1 - x_1)^2$. So, if we look at this function the minimum value is; obviously, x_1 equal to 1, x_2 equal to 1 and the function value becomes 0. So, the optimal solution for this Rosenbrock function is x_1 equal to 1, x_2 equal to 1 and at this optimal point the function value is 0. So, take the gradient, so we have evaluated the gradient.

(Refer Slide Time: 18:30).



MATLAB Optimization Toolbox: fminunc

Step-1

You first have to code the objective function. Open a new M-file in the editor and type in:

```
function [f, gradf] = objfun(x)
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
gradf(1) = -400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
gradf(2) = 200*(x(2)-x(1)^2);
end
```

The first argument f is the value of the function at the input vector x . The 2nd output argument, $gradf$, is the gradient vector of f (column vector), evaluated at x

Save the file under any name - here we choose `objfun.m`. Note all MATLAB functions will have extension `.m`.

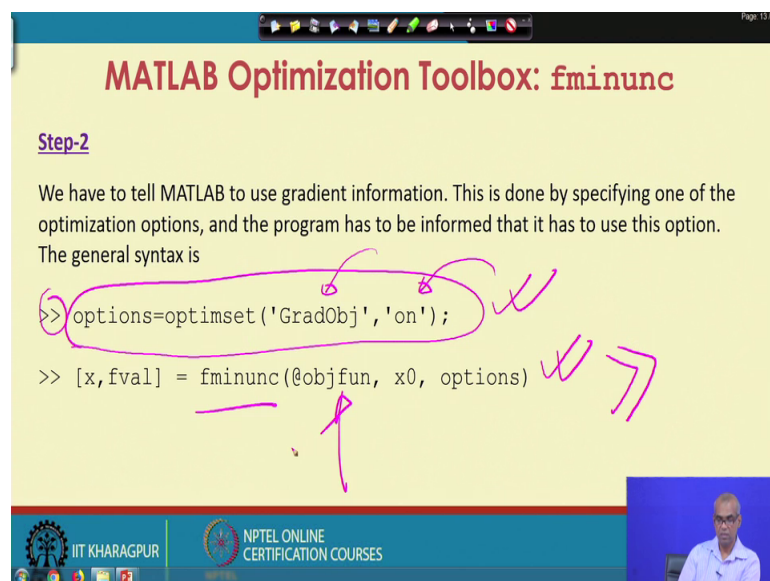
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in the first step you have to code the objective function. So, open a new M-file in the editor and you have to tie these lines. First line is the definition of the function file this is how we write the function file function then f grad f these are the 2 values that will be returned by the function objfun and x is the input argument to the objfun function.

So, here the Rosenbrock function is coded note that it is 2 dimensional function x_1 and x_2 . So, the vector x has 2 component x_1 and x_2 $x_1^2 - 1$ and $x_2 - x_1$ equal to $x_1^2 - 1$ $x_2 - x_1$ equal to x_2 . So, we write the Rosenbrock we code the Rosenbrock function then the 2 components $\frac{df}{dx_1}$ and $\frac{df}{dx_2}$ are coded as grad f 1 and grad f 2. So, that is all if you want you can write n here.

So, the first argument f is the value of the function at the input vector x, the second output argument grad f is the gradient vector of f which is again column vector evaluated at x. Now you save the file under any name here we choose objfun dot m all MATLAB functions will have extension dot m.

(Refer Slide Time: 20:48)



The slide is titled "MATLAB Optimization Toolbox: fminunc" and is labeled "Step-2". It explains that to use gradient information, one must specify optimization options. The general syntax is shown as:

```
>> options=optimset('GradObj','on');  
>> [x,fval] = fminunc(@objfun, x0, options)
```

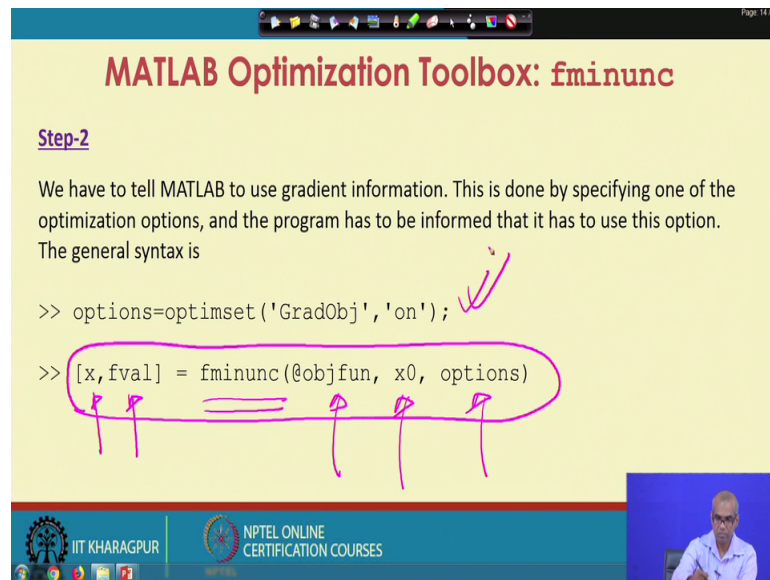
Handwritten pink annotations highlight the 'GradObj' option in the first line and the 'options' argument in the second line. The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a presenter in the bottom right corner.

Now in the step two we have to tell MATLAB that the solver fminunc has to use the gradient information. This is done by specifying one of the optimization options and the program has to be informed that it has to use this option.

So, the general syntax is options equal to optimset gradobj on; that means, we are supplying gradient and a solver has to use the supplied gradient information. This means

that this symbol represents the comment line. So, you can type it in comment line or you can create another m file give it some other name and write this statement and then call the solver fminunc to solve the problem to minimize the function written in objfun.

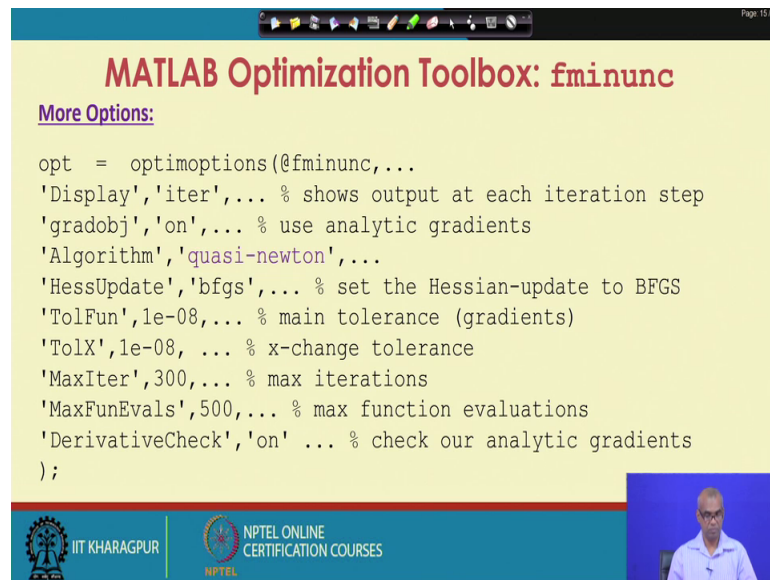
(Refer Slide Time: 22:25)



The slide is titled "MATLAB Optimization Toolbox: fminunc" and is labeled "Step-2". It explains that to use gradient information, one must specify optimization options. The general syntax is shown as two lines of MATLAB code: `>> options=optimset('GradObj','on');` and `>> [x,fval] = fminunc(@objfun, x0, options)`. Handwritten pink annotations include arrows pointing to the 'on' in the first line, a circle around the entire second line, and arrows pointing to the function handle, initial guess, and options arguments in the second line. The slide footer includes the IIT Kharagpur and NPTEL Online Certification Courses logos, and a small video inset of a presenter.

So, we are already familiar with the basic sequence basic syntax which is x, fval equal to the solver name, then arguments say our first argument is at then name of the objective name of the function where name of the function file where you have written the objective function as well as gradient. Second argument is the vector having initial guess and third is the options where you have given the option of using gradient information.

(Refer Slide Time: 23:28)



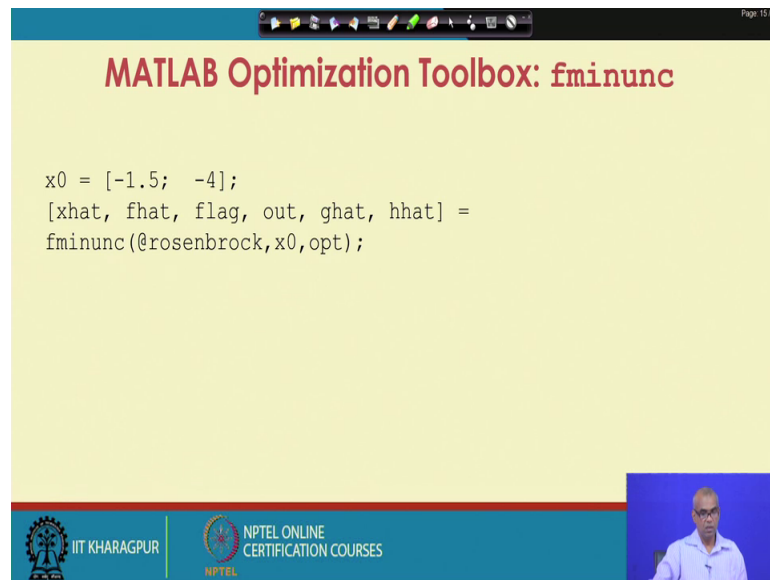
The slide is titled "MATLAB Optimization Toolbox: fminunc" in red text. Below the title, it says "More Options:" in purple. The main content is a MATLAB code snippet for setting optimization options. At the bottom left, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. At the bottom right, there is a small video inset showing a man in a white shirt.

```
opt = optimoptions(@fminunc,...
'Display','iter',... % shows output at each iteration step
'gradobj','on',... % use analytic gradients
'Algorithm','quasi-newton',...
'HessUpdate','bfgs',... % set the Hessian-update to BFGS
'TolFun',1e-08,... % main tolerance (gradients)
'TolX',1e-08, ... % x-change tolerance
'MaxIter',300,... % max iterations
'MaxFunEvals',500,... % max function evaluations
'DerivativeCheck','on' ... % check our analytic gradients
);
```

Now, so there are many other options that we can use. These options are shown here you can define these options as. So, you define as opt equal to optim options within bracket at fminunc; that means, the name of the solver and then all the options display iter; that means, it will show output at each iteration step grad obj on; that means, it will use analytic gradients. So, analytic gradients will be supplied by you in the function file.

Here we have named that as obj fun algorithm Quasi-Newton. So, I am asking the solver to use quasi Newton method to minimize the function the Rosenbrocks function using Quasi-Newton method HessUpdate means hessian update means hessian update use BFGS method. We talked about BFGS method cross region method TolFun is the tolerance main tolerance for gradients TolX, x change tolerance x is the solution vector MaxIter, 300 means maximum number of iterations is not more than 300, MaxFunEvals maximum number of function evaluations. DerivativeCheck on: So, it will check our analytic gradients. So, different options now are available and you can use all these options or some of these options.

(Refer Slide Time: 25:24)



MATLAB Optimization Toolbox: fminunc

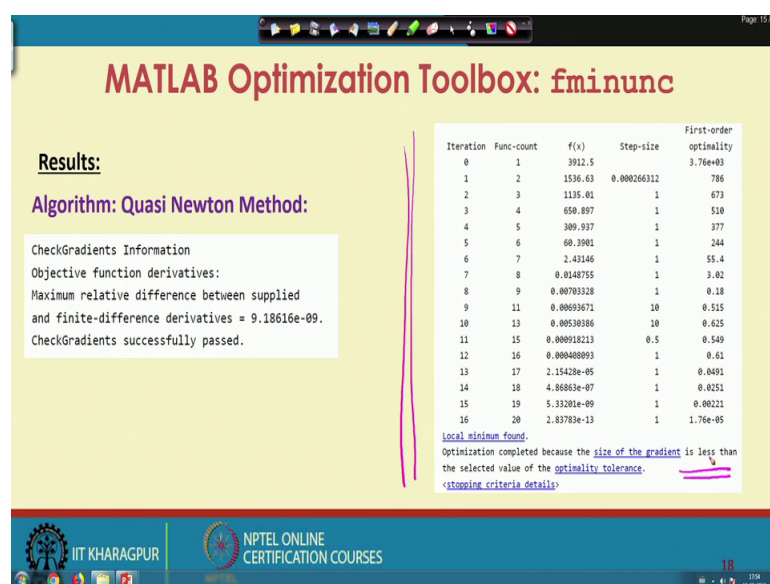
```
x0 = [-1.5; -4];  
[xhat, fhat, flag, out, ghat, hhat] =  
fminunc(@rosenbrock,x0,opt);
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us solve the Rosenbrock problem with let us say this Quasi-Newton algorithm. So, we are supplying all these options and solving the problem using Quasi-Newton algorithm. So, I supply x 0 equal to minus 1.5 and minus 4 as initial guess, then use the syntax as shown to solve the problem.

So, I have given the name as now say Rosenbrock the objfun. Now I am renaming as Rosenbrock. You can also give it any other name you want.

(Refer Slide Time: 26:17)



MATLAB Optimization Toolbox: fminunc

Results:

Algorithm: Quasi Newton Method:

CheckGradients Information
Objective function derivatives:
Maximum relative difference between supplied
and finite-difference derivatives = 9.18616e-09.
CheckGradients successfully passed.

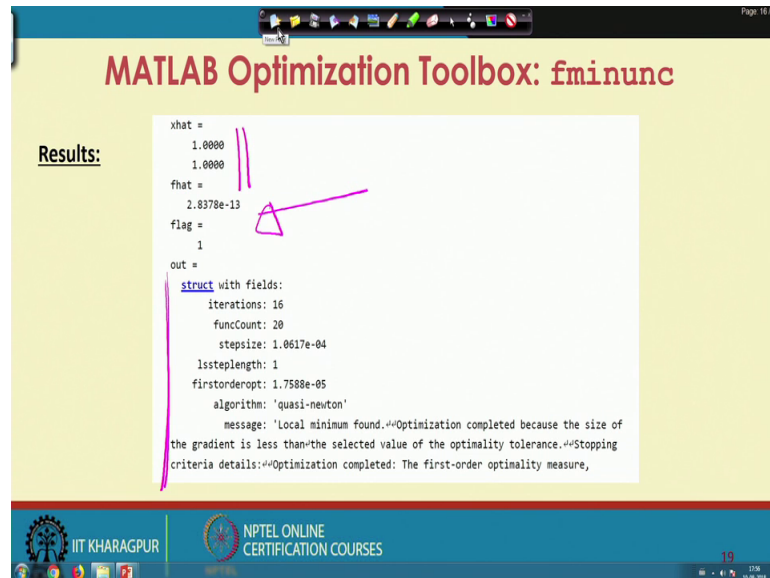
Iteration	Func-count	f(x)	Step-size	First-order optimality
0	1	3912.5		3.76e+03
1	2	1536.63	0.000266312	786
2	3	1135.01	1	673
3	4	650.897	1	510
4	5	389.937	1	377
5	6	68.3901	1	244
6	7	2.43146	1	55.4
7	8	0.0148755	1	3.02
8	9	0.00703328	1	0.18
9	11	0.00093671	10	0.515
10	13	0.00530386	10	0.625
11	15	0.000918213	0.5	0.549
12	16	0.000408993	1	0.61
13	17	2.15428e-05	1	0.0491
14	18	4.86863e-07	1	0.0251
15	19	5.33281e-09	1	0.00211
16	20	2.83783e-13	1	1.76e-05

Local minimum found.
Optimization completed because the size of the gradient is less than the selected value of the optimality tolerance.
<stopping_criteria details>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is the solution you get. So, this is a snapshot of the run of the algorithm result of the run of the algorithm. So, it first takes the gradient information's because derivative check option was on and these are the iterations wise results. So, after 16th iteration optimization is complete. The persistent iteration the size of the gradient becomes very small.

(Refer Slide Time: 27:07)



The image shows a slide titled "MATLAB Optimization Toolbox: fminunc". It displays the results of an optimization process. The results are as follows:

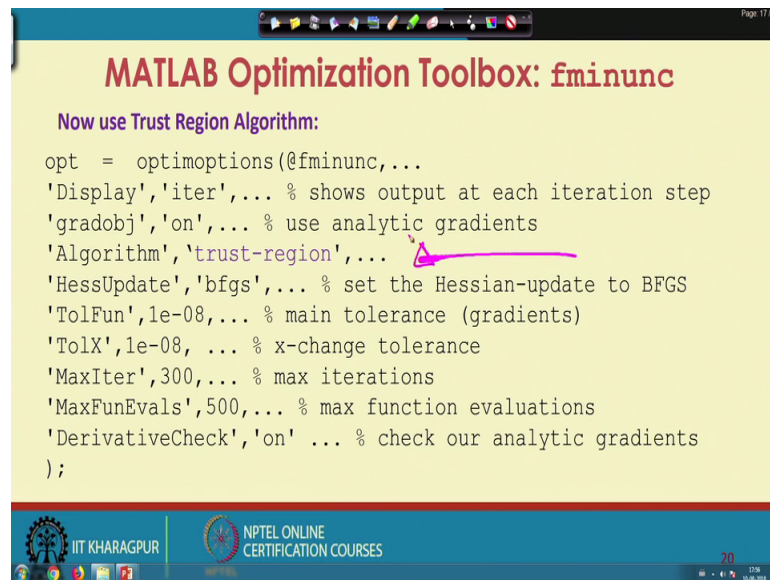
```
Results:
xhat =
    1.0000
    1.0000
fhat =
    2.8378e-13
flag =
     1
out =
 struct with fields:
    iterations: 16
   funcCount: 20
  stepsize: 1.0617e-04
  lssteplength: 1
 firstorderopt: 1.7588e-05
  algorithm: 'quasi-newton'
  message: 'Local minimum found. Optimization completed because the size of the gradient is less than the selected value of the optimality tolerance. Stopping criteria details: Optimization completed: The first-order optimality measure,
```

The slide also features the IIT Kharagpur logo and NPTEL Online Certification Courses branding at the bottom.

So, now, look at the solution that you have got 1 1 and we know that x_1 equal to 1 and x_2 equal to 1 is the solution and the function value is 0. So, we get that as 2.83 minus 10 to the power 13. So, very close very small value can be considered as 0.

So, these are the some information's, which is obtained with the structure name out, iterations equal to 16 function count 20 steps size as 1.0617 into 10 to the minus 4 first step length was 1 so on and so forth algorithm uses Quasi-Newton. So, all these details can be obtained.

(Refer Slide Time: 28:11)



MATLAB Optimization Toolbox: fminunc

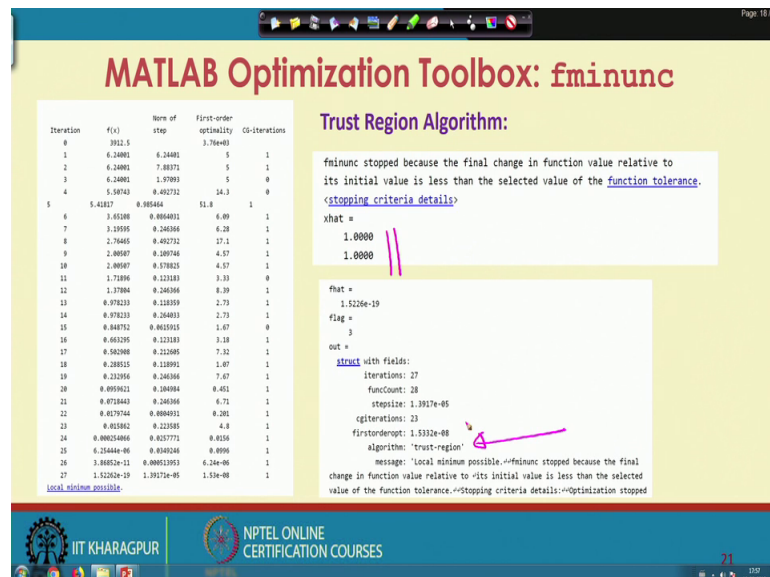
Now use Trust Region Algorithm:

```
opt = optimoptions(@fminunc,...  
'Display','iter',... % shows output at each iteration step  
'gradobj','on',... % use analytic gradients  
'Algorithm','trust-region',...  
'HessUpdate','bfgs',... % set the Hessian-update to BFGS  
'TolFun',1e-08,... % main tolerance (gradients)  
'TolX',1e-08,... % x-change tolerance  
'MaxIter',300,... % max iterations  
'MaxFunEvals',500,... % max function evaluations  
'DerivativeCheck','on' ... % check our analytic gradients  
);
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us now solve the same problem using Trust Region method. So, you just change the option here algorithm, it was Quasi-Newton method. Now I change the algorithm to Trust Region. So, now, I am telling the solver that the algorithm that needs to be use is trust region everything else remains unchanged. So, this is the result.

(Refer Slide Time: 28:59)



MATLAB Optimization Toolbox: fminunc

Iteration	f(x)	Norm of step	First-order optimality	CG-iterations
0	3912.1		3.76e+03	
1	6.24081	6.24081	5	1
2	6.24081	7.88371	5	1
3	6.24081	1.97993	5	0
4	5.59743	0.492752	14.3	0
5	5.41827	0.385464	51.8	1
6	3.85180	0.880481	6.89	1
7	3.18595	0.246366	6.28	1
8	2.76485	0.492752	17.1	1
9	2.80687	0.387466	4.57	1
10	2.80687	0.378235	4.57	1
11	1.73896	0.123383	3.33	0
12	1.37886	0.246366	8.39	1
13	0.978233	0.135959	2.73	1
14	0.978233	0.304083	2.73	1
15	0.884752	0.4615915	1.67	0
16	0.683295	0.123383	3.18	1
17	0.582968	0.222665	7.32	1
18	0.289515	0.135959	1.87	1
19	0.232956	0.246366	7.67	1
20	0.0959621	0.284984	0.451	1
21	0.0753443	0.246366	0.71	1
22	0.0279744	0.880481	0.281	1
23	0.015862	0.223585	4.8	1
24	0.000254866	0.825771	0.8256	1
25	6.2644e-08	0.804826	0.8996	1
26	3.8052e-13	0.00033953	6.10e-06	1
27	1.5226e-19	1.39376e-05	1.53e-08	1

Trust Region Algorithm:

fminunc stopped because the final change in function value relative to its initial value is less than the selected value of the function tolerance.

<stopping criteria details>

```
xhat =  
1.0000  
1.0000
```

```
fhat =  
1.5226e-19  
flag =  
3  
out =  
struct with fields:  
  iterations: 27  
  funcCount: 28  
  stepsize: 1.3917e-05  
  cgIterations: 23  
  firstorderopt: 1.5331e-08  
  algorithm: 'trust-region'  
  message: 'Local minimum possible. fminunc stopped because the final change in function value relative to its initial value is less than the selected value of the function tolerance.'<stopping criteria details>: Optimization stopped
```

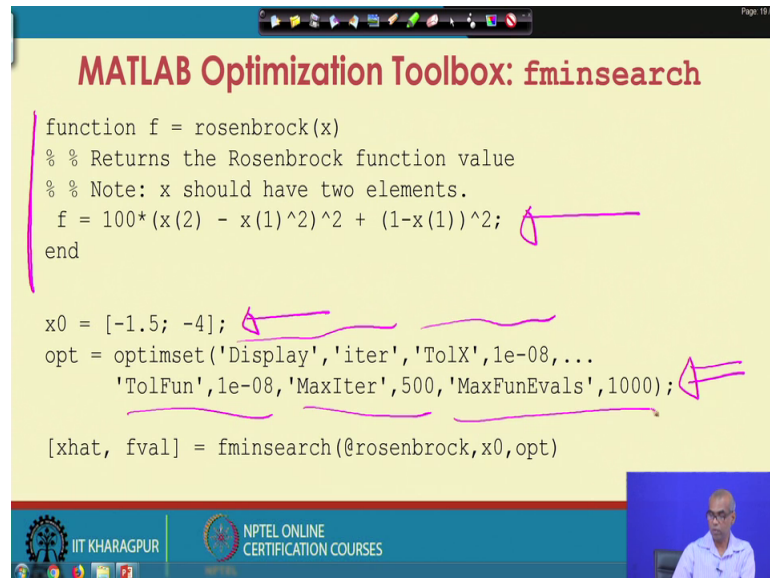
Local minimum possible.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now after 27 iterations we obtain the minimum and minimum again is obtained as x 1 equal to 1 x 2 equal to 1. The function value is 1.5226 into 10 to the power minus 19

which is very small. The details are available in the structure called out 27 iterations were required function count is 28 Trust Region algorithm was used.

(Refer Slide Time: 29:53)

A slide titled "MATLAB Optimization Toolbox: fminsearch" showing MATLAB code for the Rosenbrock function. The code defines the function, sets initial values and options, and calls fminsearch. Handwritten pink annotations highlight specific parts of the code. The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

```
function f = rosenbrock(x)
% % Returns the Rosenbrock function value
% % Note: x should have two elements.
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
end

x0 = [-1.5; -4];
opt = optimset('Display','iter','TolX',1e-08,...
'TolFun',1e-08,'MaxIter',500,'MaxFunEvals',1000);

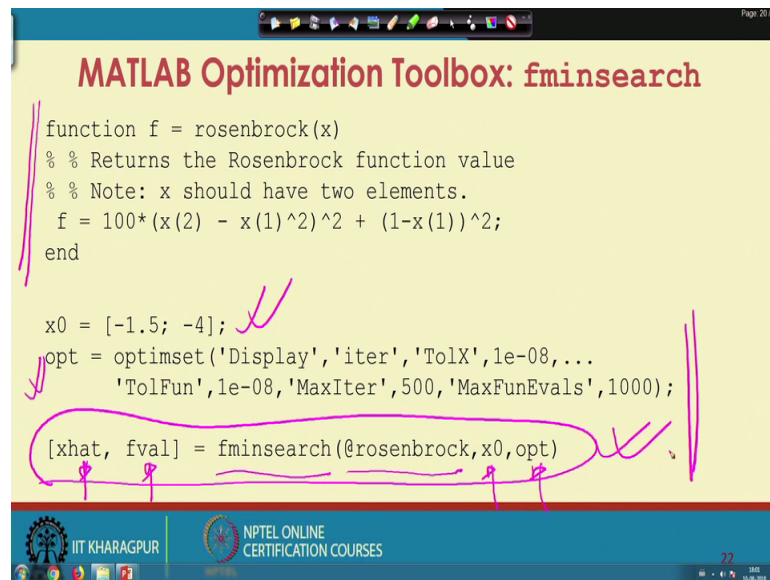
[xhat, fval] = fminsearch(@rosenbrock,x0,opt)
```

Now, let us look at the use of fminsearch. Once you are familiar with use of any solver you can easily understand the use of other solvers. For example, as of now we have seen how fminunc works. Now we are talking about fminsearch, fminsearch does not use gradient information. In fact, it implements simplest method to solve an unconstrained optimization problem.

But you will see that the basic syntax that will use are very similar. Again we define a function file let us call Rosenbrock where I quote the Rosenbrock function. He does not use gradient information, so, no need to supply the gradient information. Initial guess is given as x 0 vector same starting point I use minus 1.5 minus 4. Now let us say I give these options. So, opt equal to optimset display iter; that means, each iterations will be displayed.

X tolerance is 10 to the power minus 8, function tolerance 10 to the power minus 8, maximum iterations 500, maximum function evaluations 1000. So, once these are defined in the options.

(Refer Slide Time: 31:58)



The slide displays MATLAB code for the fminsearch function. The code defines a function 'rosenbrock(x)', sets an initial point 'x0', and uses 'fminsearch' to find the minimum. Handwritten pink annotations highlight the function definition, the initial point, the options structure, and the final function call. The slide also features the IIT Kharagpur and NPTEL logos at the bottom.

```
function f = rosenbrock(x)
%% Returns the Rosenbrock function value
%% Note: x should have two elements.
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
end

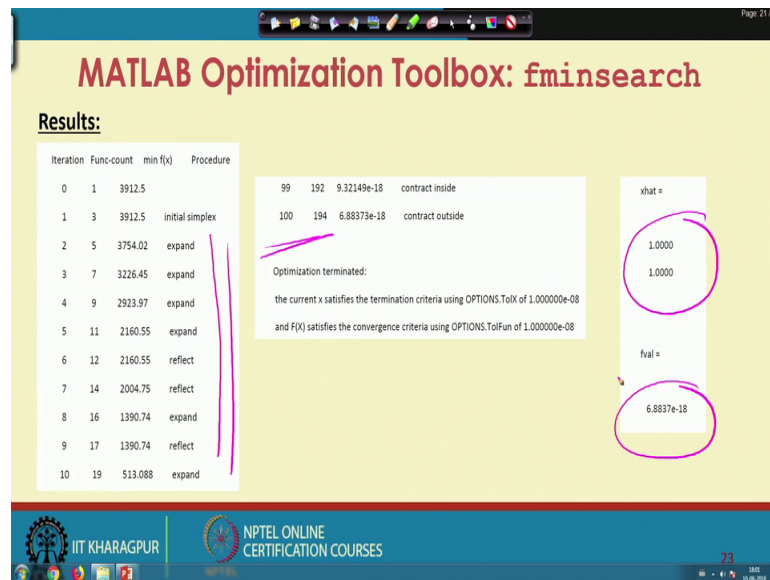
x0 = [-1.5; -4];
opt = optimset('Display','iter','TolX',1e-08,...
'TolFun',1e-08,'MaxIter',500,'MaxFunEvals',1000);

[xhat, fval] = fminsearch(@rosenbrock,x0,opt)
```

We call fminsearch using this syntax xhat fval, xhat returns the solution optimal solution, fval is the function value at optimal solution, fminsearch is the name of the solver at rosenbrock. So, this is the function file where you have coded the function x 0 is the initial starting point, initial vector is a column vector and opt is the options that we are specified.

So, you write these statements as an M-file let us call them file as Rosenberg dot m then you either type in these in the common mode both these 3 lines or write another M-file where you write these three lines and then hit the run button and you will get this solution.

(Refer Slide Time: 33:08)



Note, it is using simple x method, so it is talking about expansion of simplex, contraction of simplex, reflection all these things.

We have gone up to 100 iterations and the solution is again is obtained as x_1 equal to 1, x_2 equal to 1 the function value is 6.8837×10^{-18} which is extremely small can be considered as 0. So, we learn how to make use of MATLAB optimization toolbox. In fact, 2 function files 2 functions from MATLAB optimization toolbox namely fminunc and fminsearch. Both are used for solutions of unconstrained minimization problem. You can solve for maximization of problem simply by taking negative of f. With this, we stop our lecture number 35 here.