Lecture 3-4: MATLAB - parallel, afternotes

Optimization

Heikki Apiola

April 18, 2019

Aalto University
juha.kuortti@aalto.fi, heikki.apiola@aalto.fi

Optimization

Steps for solving

1. Choose an optimization solver.
2. Create an objective function, typically the function you want to minimize.
3. Create constraints, if any.
4. Set options, or use the default options.
5. Call the appropriate solver.
   For a basic nonlinear optimization example, see Solve a Constrained Nonlinear Problem.

fmincon, Steps for solving

fun: function to be minimized "objective function" x0: starting point: vector or n-column matrix A*x <= b: Linear inequality constraint, A matrix, b vector Aeq*x = beq: Linear equality constraint Aeq matrix, beq vector lb,ub : Lower bound, Upper bound: scalars, vectors or matrices nonlincon: Nonlinear constraints: function »options=optimoptions('fmincon') Creates struct with defaults... to be modified according to needs.

Continued

Use empty braces [] for missing arguments.

The objective function fun for all optimization routines is of the form

(1) fun=(x) f(x(1),x(2), ..., x(n)).

Thus fun is a function of one vector argument. The input x can also be an m×n matrix. Therefore the definition can also be given in the vectorized form:

(2) fun=(x) f(x(:,1),x(:,2),...,x(:,n)).

Both forms are equally good for the optimization routine, but the latter can also be used for plotting, tabulation and experimantation with several starting points etc. Note: In the 2 variable case one could also do as follows:

f=@(x,y)g(x,y) objfun=@(x) f(x(1),x(2))

Rosenbrock's function on a circle

$$r(x, y) = 100(y - x^2)^2 + (1 - x)^2 . x^2 + y^2 \leq 1$$

Code for **objective function**:

```
rosenbrock=@(x)100*(x(:,2)- x(:,1).^2).^2 + (1 ...
- x(:,1)).^2;
```

For optimization routines (here fmincon) it would be as good to write:

```
rosenbrock1=@(x)100*(x(2)- x(1)^2)^2 + (1 - x(1))^2
```

Here we have only inequality constraint, hence the **constraint function's** 2^{nd} return-value should be an empty matrix, hence we write:

Run

```
function [cineq,ceq] = unitdisk(x)
cineq = x(1)^2 + x(2)^2 - 1;
ceq = [ ];
```

Note: Rosenbrock's function is a standard test function in optimization. It has a unique minimum value of 0 attained at the point [1,1]. Finding the minimum is a challenge for some algorithms because the function has a shallow minimum inside a deeply curved valley. The solution for this problem is **not at the point [1,1]** because that point **does not satisfy the constraint**.

1. Create options that choose **iterative display** and the **interior-point algorithm**

```
options = optimoptions(@fmincon,...
    'Display','iter','Algorithm','interior-point');
```

2. Run the fmincon solver with the options structure, reporting both the location x of the minimizer and the value fval attained by the objective function.

```
[x,fval] = fmincon(rosenbrock,[0 0],...
    [],[],[],[],[],[],@unitdisk,options)
```

   The six sets of empty brackets represent optional constraints that are not being used in this example. See the fmincon function reference pages for the syntax.

Example continued

MATLAB outputs a table of iterations and the results of the optimization. Local minimum found that satisfies the constraints. Optimization completed because the objective function is non-decreasing in feasible directions, to within the selected value of the function tolerance, and constraints are satisfied to within the selected value of the constraint tolerance.