INFORMATION, LAB 2

TMA946 Applied optimization / MAN280 Optimization

Michael Patriksson

26 februari 2003

Preparations

In Part II of the exercise, there are some exercises which we recommend that you prepare by formulating the KKT conditions in advance. If you feel a slight anxiety in the proximity of computers, you may want to take a look at MATLAB's `optdemo` in advance.

Part I: Unconstrained optimization

In this part of the lab, you will solve some unconstrained problems using steepest descent and Newton's method. The latter exists in three different versions, (1) with a unit step length (the classic version), (2) Newton's "modified" method, which includes a line search, and (3) the Levenberg–Marquardt modification, where diagonal terms are added, if necessary, such that the eigenvalues of the resulting matrix are positive. The methods are implemented in MATLAB. The purpose of this lab is to graphically illustrate the methods in order to give an insight into their behaviour.

To start, download the tar-file from the course homepage, and follow the instructions given there. Move to the directory `ILP` (by giving the command `cd ILP`) and start MATLAB by simply typing `matlab`. Once MATLAB starts up, type `ilpmeny` in the command window.

The lab is menu-driven and mostly self-explaining. The following selections may be done:

Setting	Default value
Starting point	0 0
termination criterion	Gradient length
Function to be minimized	Function 1
Maximal number of iterations	200
Printing of iteration data	On
Method	Steepest descent

You may choose to take 1, 10 or 100 iterations at a time and follow the algorithm search path in the graph.

Exercises

In all these exercises, the function is to be **minimized**.

1. Study **function 1**

$$f(x_1, x_2) := 2(x_1 + 1)^2 + 8(x_2 + 3)^2 + 5x_1 + x_2.$$

   (a) Solve the problem by using steepest descent and Newton's method (unit step). Start at the points $(10, 10)^{\mathrm{T}}$ and $(-5, -5)^{\mathrm{T}}$ as well as in some starting point of your own choice. Toward which point do the methods converge? How many iterations are required?

   (b) Is the point obtained an optimal point (globally or locally)?

   (c) Why does Newton's method always converge in *one* iteration?

2. Study **function 2** (Rosenbrock's function)

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

   (a) Solve the problem by using steepest descent and Newton's method (all versions). Start at the point $(-1.5, -1)^{\mathrm{T}}$. Towards which point does the methods converge? How many iterations are required?

   (b) Is the function convex? Is the obtained point a global optimum?

   (c) Choose some starting points on your own to study the methods' behaviour.

3. Study **function 4**

$$f(x_1, x_2) := x_1^3 + 2x_2^3 - 27x_1 - 6x_2.$$

(a) Start at the point $(0,0)^{\mathrm{T}}$ and solve the problem using steepest descent and Newton's method (unit step). Why does not Newton's method work? Try the Levenberg–Marquardt modification and study that method's behaviour.

(b) Start at some arbitrarily chosen points. How many stationary points do you find? Which kinds of stationary points?

Those who wish to play further may test the other functions. They are:

(3) $f(x_1,x_2) := -5e^{-\frac{x_1^2+x_2^2}{10}}+4e^{-\frac{x_1^2+x_2^2}{100}}-5^{-\frac{(x_1-5)^2+(x_2+4)^2}{10}}-5^{-\frac{(x_1+4)^2+(x_2-5)^2}{10}}-4^{-\frac{(x_1+4)^2+(x_2-5)^2}{100}}$

(5) $f(x_1,x_2) := -4e^{-\frac{(x_1+2)^2+(x_2+1)^2}{10}}+4e^{-\frac{(x_1+2)^2+(x_2+1)^2}{100}}+0.01((x_1+2)^2+(x_2+1)^2)+0.01x_1$

(6) $f(x_1,x_2) := (x_1^3-x_2)^2+2(x_1-x_2)^4$

(7) $f(x_1,x_2) := -5\cos(0.2(1+\frac{x_2^2-0.5}{x_1^2+0.5})+0.001x_2^4+0.003x_1^4+2x_1$

(8) $f(x_1,x_2) := 2(x_2-x_1^3)^2+0.1(x_1+2)^2+0.5x_2^2+x_1^2x_2^2$

Part II: Constrained optimization; MATLAB:s Optimization Toolbox

In this part of the lab, you are to use MATLAB's optimization routines to solve some nonlinear problems.

Note! This part of the lab should be prepared by formulating the KKT conditions for the exercise problems. We have created an example to show how Matlab's constrained minimization, `fmincon`, works. In order to understand the command, it is helpful to read Matlab's help about it (type `help fmincon`).

Example:

$$\begin{array}{lllll} \min & f(x) := 2x_1^2+x_2^2 & & & \\ \text{s.t.} & x_1 & & \leq & 1 \\ & x_1^2 & + \quad x_2 & \geq & 2 \end{array}$$

The objective function is given in the file `testf.m`, and the constraints in `testg.m`. To solve the other problems, you need to create similar files.

We now call fmincon by using the command

```
[x,y,ef,output,lambda] = fmincon('testf',[0 0],[],[],[],[],[],[],'testg')
```

which means that we wish to minimize the function testf, starting at $(0,0)^{\mathrm{T}}$, using no linear inequality constraints, no linear equality constraints, and no bounds on the variables. (We treat $x_1 \leq 1$ as a general nonlinear constraint.) Once the solver is done, `x` will contain the optimal point, `y` the optimal objective value, and `lambda` the Lagrange multipliers. For the other variables, see Matlab's help files. For example, to see the Lagrange multipliers for the nonlinear inequality constraints, write

`lambda.ineqnonlin`.

Exercises

1. Given is the problem

$$\begin{array}{lrcrcl} \max & f(x) := x_1 - 2x_1^2 + 2x_2 - x_2^2 + x_1x_2 & , \\ \text{s.t.} & x_1^2 & - & x_2 & \leq & 0, \\ & 2x_1 & - & x_2 & \geq & 0. \end{array}$$

   (a) Solve the problem using `fmincon`.

   (b) State the KKT conditions, examine the convexity of the problem and verify that the obtained solution is a *global minimum*.

2. Given is the problem

$$\begin{array}{lrcrcl} \min & f(x) := x_1 & & & & , \\ \text{s.t.} & (x_1-1)^2 & + & (x_2+2)^2 & \leq & 16, \\ & x_1^2 & + & x_2^2 & \geq & 13. \end{array}$$

   Solve the problem from at least five starting points. Describe what happens. Which point is the best one? Can you guarantee that this is a *global minimum*? Fun points to try are $(1,1)^{\mathrm{T}}, (0,0)^{\mathrm{T}}, (3.7,0)^{\mathrm{T}}$ and $(-1,-1)^{\mathrm{T}}$.

Part III: Constrained optimization: penalty methods

Consider the problem to

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & g(x) \leq 0^m, \end{array}$$

where f and g are continuously differentiable functions. Penalty methods are generally of one of two different kinds: *exterior* and *interior* penalty methods, depending on if the methods generally give an infeasible of strictly feasible sequence of iteration points. We have implemented one method of each kind in MATLAB.

In order to run the programs, you should move from the directory `ILP` to the directories `ILP/epa` (exterior penalty algorithm) or `ILP/lipa` (interior penalty, or interior point, algorithm for linear programming). Both algorithms are started by typing `go` in Matlab's command window.

Note that the problems are given with the constraints on "$\leq$"-form, while Nash–Sofer describes the methods using the "$\geq$"-form.

The *exterior* penalty method ("penalty method" in Nash-Sofer) works with the relaxation

$$\underset{x\in\Re^n}{\text{minimize}}\ f(x)+\rho_k\psi(x),$$

where $\rho_k > 0$ and $\rho_k \to +\infty$ when $k \to +\infty$, and the penalty function is the quadratic function

$$\psi(x) := \sum_{i=1}^{m} \left(\max\{0, g_i(x)\}\right)^2.$$

The *interior* penalty method ("barrier method" in Nash–Sofer) works with the relaxation

$$\underset{x\in\Re^n}{\text{minimize}}\ f(x)+\mu_k\phi(x),$$

where $\mu_k > 0$ and $\mu_k \to 0$ when $k \to +\infty$, and where the penalty function is the function

$$\phi(x) := -\sum_{i=1}^{m} \log(-g_i(x)).$$

In order to avoid numerical problems one usually lets the sequences ρ_k and μ_k converge slowly.

Description of the interface

After choosing the example from the drop-down list in the lower-left part of the window, press the "Load" button. In the left window you will see the level sets of the objective function (you can think of them as a topographic map), the directions the *negative* gradient, as well as the curves constraining the feasible set. The coordinates of the current iteration point can be read below the left window.

After adjusting the desired penalty value, press the "Optimize" button. The right window will show the level sets of the penalised function, exactly as the algorithm would "see" it, were it not near-sighted. The current iteration point is plotted in the right window (pink "x" cross); the left window will contain the optimization "path", showing the progress of the algorithm (pink curve).

Note! In our implementation of EPA we solve the penalised problem using a gradient algorithm to obtain a globally optimal solution. Instead, one can perform only a few iterations of the gradient algorithm.

In IPA, we perform *only one* iteration of the modified Newton method (with an Armijo line search). We show the global minimum point in the right window using a red "o" circle; its evolution as the penalty parameter changes (so-called "central path") is shown in the left window as a red line.

Hint! You can change the starting point for the algorithm by modifying the variables `x1_start` and `x2_start` in the `.m`-file, corresponding to the example you solve. Even more, you can add your own problems by providing a corresponding `example*.m` file!

Exercises

There are four nonlinear problems, two convex (`example_nl{01,02}.m`), and two non-convex (`example_nl{03,04}.m`), as well as three linear problems (`example_lin[01-03].m`); you can find the problem formulations in the Appendix.

1. Using the interior point method, solve the LPs 01–03. Do we always find an optimal extreme point (problem 03)? Notice how the algorithm follows closely the central path and goes "directly" to the global minimum point (i.e., it skips visiting the extreme points), if you change the penalty parameter smoothly. Compare with the Simplex method.

2. Change the directory to `ILP/kkt` and type `go` at the Matlab prompt. Find the KKT points of the nonlinear problems (`example_nl[01-04].m`). Are the KKT conditions sufficient for the global optimality (problem 03)? Are they necessary (problem 04)?

   Hint: Matlab does not calculate the Lagrange multipliers exactly. This hint is especially important for the problem 04.

3. Using the exterior penalty algorithm, solve the nonlinear and linear problems. Can you get different "optimal" solutions by changing the penalty parameter in a different manner or by starting from different

points (problem 03)? Why does the algorithm converge so slowly in the problem 04 (hint: KKT)?

Appendix

Linear problems

`example_lin01.m`

$$\begin{aligned} &\min x_1 + 3x_2, \\ &\text{s.t.} \begin{cases} x_1 + 2x_2 \geq 2, \\ x_1 - 3x_2 \leq 2, \\ -x_1 + 3x_2 \leq 12, \\ x_1, x_2 \geq 0, \end{cases} \end{aligned}$$

`example_lin02.m`

$$\begin{aligned} &\min x_1 + 3x_2, \\ &\text{s.t.} \begin{cases} 1x_1 + 1/10x_2 \geq 1, \\ 1/2x_1 + 1/9x_2 \geq 1, \\ 1/3x_1 + 1/8x_2 \geq 1, \\ 1/4x_1 + 1/7x_2 \geq 1, \\ 1/5x_1 + 1/6x_2 \geq 1, \\ 1/6x_1 + 1/5x_2 \geq 1, \\ 1/7x_1 + 1/4x_2 \geq 1, \\ 1/8x_1 + 1/3x_2 \geq 1, \\ 1/9x_1 + 1/2x_2 \geq 1, \\ 1/10x_1 + 1x_2 \geq 1, \\ x_1 \leq 20, \\ x_2 \geq 0. \end{cases} \end{aligned}$$

`example_lin03.m`

$$\min x_2,$$
$$\text{s.t.} \begin{cases} 1x_1 + 1/10x_2 \geq 1, \\ 1/2x_1 + 1/9x_2 \geq 1, \\ 1/3x_1 + 1/8x_2 \geq 1, \\ 1/4x_1 + 1/7x_2 \geq 1, \\ 1/5x_1 + 1/6x_2 \geq 1, \\ 1/6x_1 + 1/5x_2 \geq 1, \\ 1/7x_1 + 1/4x_2 \geq 1, \\ 1/8x_1 + 1/3x_2 \geq 1, \\ 1/9x_1 + 1/2x_2 \geq 1, \\ 1/10x_1 + 1x_2 \geq 1, \\ x_1 \leq 20, \\ x_2 \geq 0. \end{cases}$$

Nonlinear problems

`example_nl01.m`

$$\min x_1^2 + x_2^2,$$
$$\text{s.t.} \begin{cases} x_1 \geq 2, \\ x_2 \geq 1, \\ 1/2x_1 + 1/4x_2 \leq 2. \end{cases}$$

`example_nl02.m`

$$\min x_1^2,$$
$$\text{s.t.} \begin{cases} x_1 \geq 2, \\ x_2 \geq 1, \\ 1/2x_1 + 1/4x_2 \leq 2. \end{cases}$$

`example_nl03.m`

$$\min x_1 \sin(x_1) + x_2 \sin(x_2),$$
$$\text{s.t.} \begin{cases} x_1 \geq 1/3, \\ x_2 \geq 3/4, \\ x_1 - \sin(x_2) \geq 0, \\ x_1^2 + x_2^2 \leq 5. \end{cases}$$

`example_nl04.m`

$$\min (x_1 + 1)^2 + 1/2 x_2^2,$$
$$\text{s.t.} \begin{cases} x_1 \leq 3, \\ x_2 \geq 0, \\ 1/8 x_1^3 - x_2 \geq 0. \end{cases}$$