Information on Exercise 2, nonlinear programs and software
MVE165 Applied Optimization

April 29, 2008

The purpose of these computer exercises is to get more familiar with using software for computing the solution of nonlinear programs. This will be helpful when you work with the projects, and hopefully also in your future career. You will use the following software;

- Matlab optimization toolbox,
- Clp.

Hopefully you are familiar with the Matlab optimization toolbox and Clp from the first computer exercise.

Following the exercises, you will see the different solvers with their strengths and drawbacks. Note that the algorithms you will evaluate are called local optimization algorithms. This means that they try to find local optimal solutions.

The examination is preferably oral and accomplished at the computer laboration occasion on the 6:th of May, 17–19.

Preparation

Read chapters 18–19 in Taha and the notes from Lectures 10–12.

Exercise 1 – Unconstrained optimization

In this exercise, you will use a Matlab GUI to solve two unconstrained optimization problems. In the GUI, you can switch between the steepest descent method and Newton's method. Newton's method is implemented in three different versions; with a unit step length (classic), with a line search, and with the Levenberg-Marquardt modification (adding a positive diagonal matrix to the Hessian matrix).

- Download the zip-file `nlp_2008.zip` from the course web page, and unzip it in a suitable folder.

  ```
  > unzip nlp_2008.zip
  ```

  Move to the directory LAB1. Start matlab in that directory and type `ilpmeny` in the Matlab command window to access the GUI.

Answer the questions below, and motivate your answers.

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 to minimize f over $\Re^2$ using steepest descent and Newton's method (unit step). Start at the points $(10, 10)^{\mathrm{T}}$ and $(-5, -5)^{\mathrm{T}}$. 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 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 to minimize f over $\Re^2$ using steepest descent and Newton's method (unit step). Start at the point $(-1.5, -1)^{\mathrm{T}}$. Towards which point does the method converge? How many iterations are required?

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

Exercise 2 – Quadratic programming

A quadratic problem (QP) has a quadratic objective function and linear constraints. Consider the quadratic problem (QP1):

$$\begin{aligned} \min f := \tfrac{1}{2}(x_1 - 1)^2 + \tfrac{1}{2}(x_2 - 5)^2 \\ -2x_1 + x_2 \leq 2 \\ -x_1 + x_2 \leq 3 \\ x_1 \leq 3 \\ x_1, x_2 \geq 0 \end{aligned}$$

Solve (QP1) graphically.

State the KKT-conditions for (QP1).

Is x^* a KKT-point, i.e., does it satisfy the KKT conditions. Is x^* a (global) optimal solution?

In the Matlab Optimization Toolbox, there is a driver routine called `quadprog` for solving quadratic problems. It has an interface which is simpler than the one for solving a general constrained optimization problem. The reason for this is that a QP can be written using matrices solely:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}x^{\mathrm{T}}Hx + x^{\mathrm{T}}d, \\ \text{subject to} \quad & Ax \leq b, \\ & A_{eq}x = b_{eq}, \\ & x_l \leq x \leq x_u \end{aligned}$$

Try `help quadprog`. The above problem is solved using

```
>> [x, fval] = quadprog(H, d, A, b, Aeq, beq, xl, xu);
```

As with `linprog`, you can set options for the solver using the structure `options`. Try `help optimoptions`, and `help optimset`. `quadprog` uses two methods for solving (QP): medium-scale method and large-scale method. The medium-scale method is an active-set method. It works similar to the simplex method: the simplex method moves between vertices of the polyhedron (extreme points) and the active-set method for QPs moves along the edges of the polyhedron. It is not restricted to extreme points (and actually not even edges as it may happen that the optimum is unconstrained).

If the constraints in the QP is only lower and upper bounds, or if it is only equality constraints, then `quadprog` uses the large-scale method. It is a trust-region method based on the Newton method. In each iteration, a linear system involving the Hessian is solved iteratively. This method is much faster than the active set method.

Solve (QP1) using `quadprog`.

Clp can also solve quadratic problems. It uses a barrier method (also called interior-point method), which means that the inequality constraints are penalized using logarithmic functions and the iterates move in the interior of the feasible domain (compared to the active-set method which moves along the edges). Clp can solve quite large quadratic problems.

You will compare the performance of `quadprog` against Clp on a few difficult QPs.

- Download `qpset.zip` from the course web page, and unzip it in a suitable folder. It contains one mat-file for each problem, and each mat-file contains matrices A, A_{eq}, H and vectors d, b, b_{eq}, x_l and x_u.

To use Clp, type (in Matlab)

```
>> addpath /chalmers/sw/unsup/Clp-1.6.0
```

The driver routine is called `clp`. Try `help clp`. The matrix Q is the Hessian (here H).

Compare the performance of the solvers on the QP test set.

Exercise 3 – Nonlinear constrained optimization

In this exercise you will solve nonlinear constrained problems using the driver routine `fmincon` from the Matlab Optimization Toolbox. The (two) problems you will solve are from a test set called Hock-Schittkowski. The set was collected in 1980, and it contains about one hundred small nonlinear problems. The first problem is:

$$\text{(HS22)} \qquad \begin{aligned} \text{minimize} \quad & f(x) := (x_1 - 2)^2 + (x_2 - 1)^2, \\ \text{subject to} \quad & -x_1 - x_2 + 2 \geq 0, \\ & -x_1^2 + x_2 \geq 0, \end{aligned}$$

and the second problem is:

$$\text{(HS47)} \qquad \begin{aligned} \text{minimize} \quad & f(x) := (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4, \\ \text{subject to} \quad & x_1 + x_2^2 + x_3^3 - 3 = 0, \\ & x_2 - x_3^2 + x_4 - 1 = 0, \\ & x_1 x_5 - 1 = 0. \end{aligned}$$

Try `help fmincon` to see how to use the driver. It is recommended that you create two function files for each problem; one function file for the objective function and one for the constraints. For example, if you want to solve the first problem: create `objhs22.m` for the objective function and `conhs22.m` for the constraints. The header for the objective function should be

```
function f = objhs22(x)
```

and for the constraints

```
function [c, ceq] = conhs22(x)
```

In `objhs22`, the objective value is `f`. In `conhs22`, `c` is a vector with one component for each inequality constraint (of the form $c_i(x) \leq 0$) and `ceq` is a vector with one component for each equality constraint (of the form $ceq_i(x) = 0$). If you have linear constraints or simple bounds on the variables, you can either include them in the constraint function, or specify them as you did with `quadprog`. The full interface for `fmincon` is

```
[x, f] = fmincon(@objhs22, x0, A, b, Aeq, beq, xl, xu, @conhs22, options);
```

If you don't supply any derivatives for the objective function and the constraints, `fmincon` will use finite-difference to numerically approximate the derivatives. If you do have the gradient g of the objective function, change `objhs22` to

```
function [f, g] = objhs22(x)
```

and change options using

```
>> options = optimset('GradObj', 'on');
```

If you want to use the Jacobian of the constraints, read the help for `fmincon`.

Solve (HS22) and (HS47) using `fmincon`.

Do you find a (global) optimal solution? Are the problems equally difficult/easy to solve?

The driver `fmincon` is a Sequential Quadratic Programming (SQP) solver. In each iteration a QP is solved. In the absence of inequality constraints it is similar to the Langrangean method in chapter 18.2 of Taha. SQP is very popular for small- and medium-scale problems, since it requires few function evaluations and is very robust.