Chalmers University of Technology
University of Gothenburg
Mathematical Sciences
Optimization
Ann-Brith Strömberg

MVE165
MMG630
Applied Optimization
Exercise information
April 26, 2010

Exercise 2: Nonlinear programming and software

Introduction

The purpose of these computer exercises is to get more familiar with using software for solving nonlinear programs. This will be helpful for the assignments and hopefully also in your future career. You will use the software Matlab Optimization Toolbox and clp.

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

Examination

The exercise should be performed in groups of two or three students. The examination is preferably oral and accomplished at the computer laboration occasion on the 27:th of April at 16–18, or the 4:th, 6:th, or 11:th of May at 16–19. Students not present at this occasion must hand in a written report (not later than the 12:th of May at 17.00).

Preparation

Read Chapters 9–12 the course book and the notes of Lectures 11–13.

Exercise 2.1 – Unconstrained optimization

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

- Download the zip-file `nlplab09.zip` from the homepage and unzip it in a suitable folder:

  ```
  > unzip nlplab09.zip
  ```

  Move to the directory NLPLAB09. 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 the steepest descent and Newton's (unit step) methods. Start at the points $(10, 10)^{\mathrm{T}}$ and $(-5, -5)^{\mathrm{T}}$. Towards which point do the methods converge? How many iterations are required?
   (b) Is the point obtained an optimal point (globally or locally)? Why?
   (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 the steepest descent and Newton's (unit step, Levenberg–Marquardt, and modified) methods. Start at the point $(-1.5, -1)^{\mathrm{T}}$. Towards which point do the methods converge? How many iterations are required for the different methods?
   (b) (Where) Is **Function 2** convex? Is the point obtained a global minimum?
3. Study **Function 9:** $f(x_1, x_2) := \cos(x_1^2 - 3x_2) + \sin(x_1^2 + x_2^2)$
   (a) Solve the problem to minimize f over $\Re^2$ using the steepest descent and Newton's (unit step, Levenberg–Marquardt, and modified) method. Start at the point $(1.2, 0.5)^{\mathrm{T}}$. Towards which points do the methods converge? How many iterations are required for the different methods? Does something unexpected happen? What is the explanation?
   (b) Is **Function 9** convex? Are the points obtained global minima?

Exercise 2.2 – Quadratic programming

A quadratic program (QP) has a quadratic objective function and linear constraints. Consider the quadratic program

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

$\boxed{\text{Solve (1) graphically}}$

$\boxed{\text{State the Karush–Kuhn–Tucker conditions for (1)}}$

Is x^* a Karush–Kuhn–Tucker point, i.e., does it satisfy the Karush–Kuhn–Tucker conditions. Is x^* an (global) optimal solution?

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

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

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 (2): a medium scale and a large scale method. The medium scale method is a so called 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 quadratic programs 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 quadratic program consist of only lower and upper bounds, or if it has 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.

$\boxed{\text{Solve (1) using } \texttt{quadprog}}$

Clp can also solve quadratic programs. 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 programs.

You will compare the performance of `quadprog` against clp on a few difficult QP's.

- Download `qpset.zip` from the homepage, 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 (denoted by H in (2)).

$\boxed{\text{Compare the performance of the solvers on the QP test set}}$

Exercise 2.3 – 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 defined as

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

and the second problem is defined as

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

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 complete 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)
```

define the gradient `g` in the file `objhs22.m` and change options using

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

To utilize the Jacobian of the constraints, read the help for `fmincon`, change `conhs22` and the options analogously.

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

For (HS47), try different starting points, e.g. $(1, 1, 1, 1, 1)$, $(-1, -1, -1, -1, -1)$, and $(10, 10, 10, 10, 10)$.

Draw a graphical picture of (HS22)

Do you find global optimal solutions? Are the two problems equally difficult/easy to solve?

The driver `fmincon` is a Sequential Quadratic Programming (SQP) solver. In each iteration a quadratic program is solved. SQP is very popular for small- and medium-scale problems, since it requires few function evaluations and is very robust.