Information on Exercise 1, linear programs and software
MVE165 Applied Optimization

April 2, 2008

The purpose of these computer exercises is to get more familiar with using software for computing the solution of linear 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,
- AMPL/CPLEX,
- GLPK, and
- Clp.

The optimization toolbox in Matlab can handle linear, nonlinear unconstrained and constrained, and binary (linear) programs. There are also special purpose solvers for quadratic programs and nonlinear least squares problems. Each problem type has a driver routine (e.g. `linprog` for LP). Try `'help optim'` to see a list of the routines in the toolbox.

AMPL[1] is an algebraic modeling language for mathematical programming. CPLEX[2] is an LP/IP solver which has an interface to AMPL and Matlab. GNU Linear Programming Kit[3] (GLPK) is an LP/IP solver which has an interface[4] to Matlab. Clp[5] is a LP/QP solver which has an interface[6] to Matlab. Following the exercises, you will see the different solvers with their strengths and drawbacks.

The examination is preferably oral and accomplished at the computer laboration occasion on the 8:th of April, 17–19.

---

[1] http://www.ampl.com/
[2] http://www.ilog.com/products/cplex/
[3] http://www.gnu.org/software/glpk/
[4] http://glpkmex.sourceforge.net/
[5] https://projects.coin-or.org/Clp
[6] http://control.ee.ethz.ch/~joloef/clp.php

Preparation

Read (at least) chapters 1, 2.1–2.3, 2.4.2, and 3.1–3.3 in the book by Taha (or the corresponding material in the book by Andréasson et al.).

Exercise 1.1 – Matlab

We will start with a simple problem (LP1):

$$\begin{array}{lrcl} \text{minimize} & z = -x_1 - 2x_2, & & \\ \text{subject to} & -2x_1 + x_2 & \leq & 2, \\ & -x_1 + x_2 & \leq & 3, \\ & x_1 & \leq & 3, \\ & x_1, \quad x_2 & \geq & 0. \end{array}$$

Solve (LP1) graphically.

Implement and solve (LP1) using `linprog` in Matlab.

The problem does not need to be in standard form, however it must be in matrix form. Try `help linprog`; it can handle equality constraints (Aeq), inequality constraints (A), lower (lb) and upper bounds (ub) on variables. Of course, bounds on variables can be formulated with general linear constraints, but it is often more efficient to deal with them explicitly. This is extra important if the problem is large.

With the structure `options`, the user can influence the algorithm. Try the option `'help optimoptions'` to see a list of options. To set any of them, for example `Display` and `MaxIter`, you can generate the structure `options` by

```
>> options = optimset('Display', 'on', 'MaxIter', 100);
```

and solve using (see also `'help linprog'`)

```
>> [x,f] = linprog(c, A, b, [], [], lb, [], x0, options);
```

where `x0` refers to a starting point (which may be omitted) and `options` is set according to the following. To choose between different LP solvers (simplex and interior-point), type e.g.:

```
>> options = optimset('Simplex', 'on', 'LargeScale', 'off');
```

For this simple example, you will see no difference between the methods, but for larger, more difficult problems there is a huge difference. The simplex method implemented in `linprog` seems to be more robust than the interior-point method. However simplex in `linprog` can not handle sparse matrices, which makes it very slow for large problems.

Exercise 1.2 – AMPL

In this exercise you are given a problem to model as a linear program. You will write the model in `AMPL` and solve it with `CPLEX`.

The alloy problem:
A nonferrous metals corporation manufactures four different alloys from four basic metals. The objective is to determine the optimal product mix to maximize gross revenue while not exceeding the supply limits. The requirements are given in the table below. Formulate a linear optimization model for this problem.

Metal	Proportions of metal in alloy 1	2	3	4	Total supply of metal/day
1	0.25	0.6	0.2	0.1	5 tons
2	0.25	0.2	0.6	0.7	5 tons
3	0.25	0	0.1	0.1	1 ton
4	0.25	0.2	0.1	0.1	2 tons
Selling price of alloy/ton:	$30	$15	$25	$23	

Implement the model in AMPL and solve it with CPLEX.

In `AMPL` you work with three files; a script file (`name.run`), a model file (`name.mod`), and a data file (`name.dat`). Your variable names, parameter names, objective function and constraints are formulated in the model file. All the data (e.g. the entries in the cost vector c) are specified in the data file. The solution course is indicated in the script file.

The files `lp1.run`, `lp1.mod`, and `lp1.dat` (for solving the model (LP1)) can be downloaded from the course home page
`http://www.math.chalmers.se/Math/Grundutb/CTH/mve165/0708/`.

To solve the above model, modify these files and solve the problem by typing[7] at the prompt (presuming that your script file is named `alloy.run`):

```
ampl alloy.run
```

Note: It is easy to transform the LP into an IP. Just add `integer` to the variable declaration in `lp1.mod`:

```
var x[J] integer >= 0;
```

[7]If you would like to work at home, you can download a student version of `AMPL`.

Exercise 1.3 – Netlib

The NETLIB[8] repository contains a large collection of numerical software. There is a LP test set, which has been used extensively for benchmarking LP solvers. By today standard, the problems are small scale, but many of them are quite difficult to solve nonetheless. The test problems have been submitted by several people, and many of them are based on real applications.

- From the course web page, either download the recommended netlib examples `lpcoll.zip` (or the whole netlib repository `netlibfiles.zip`) and unzip it using the command `unzip lpcoll.zip` (or, correspondingly, `unzip netlibfiles.zip`). All these examples are in mat-format.[9] Also, download and a text file `lptest.txt` with information on the size and optimal value for each problem.

- Each mat-file contains `A, b, c, lo, hi, z0` for a problem in the following form:

$$\min z := c^{\mathrm{T}}x + z0, \qquad \text{s.t.} \quad Ax = b,\ lo \leq x \leq hi.$$

  To load a file `'file.mat'` in Matlab, type `load file.mat`.

In this exercise you will use the Matlab optimization toolbox, GLPK, CLP and Cplex to solve the Netlib test problems.

Compare the performance of the solvers on the Netlib LP test set.

Measure the computational time and whether the correct optimum is found. You don't have to try all problems. Pick a few of varying size. Can you draw any conclusions on which solver to use?

To use GLPK, type (in Matlab)

```
>> addpath /chalmers/sw/unsup/glpk-4.21/mex
```

The driver routine is called `glpk`. Try `help glpk`. When you only have equality constraints as in this case, the vector `ctype` should be all 'S', according to:

```
>> for j=1:size(A,1)
>>   ctype(j) = 'S';
>> end
```

To use Clp, type (in Matlab):

[8] http://www.netlib.org/

[9] http://www.math.ufl.edu/~hager/coap/

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

The driver routine is called `clp`. Try `help clp`. (`Q` is the Hessian of the objective function. Set it to `'[ ]'`.)

Finally, there is also an interface[10] to Matlab for the solver Cplex. To use it, type (in Matlab):

```
>> addpath /chalmers/sw/unsup/cplexmex/dist
```

It's interface is similar to GLPK, but ctype should now be a vector of 'E':s and it must be a column vector, so:

```
>> for j=1:size(A,1)
>>   ctype(j) = 'E';
>> end
>> ctype = ctype(:);
```

The driver routine is called `cplexmex`. Try `help cplexmex` and `help cplexmexparams`.

[10] `http://www.dii.unisi.it/~hybrid/tools/mex/downloads.html`