Optimization, Programming Assignment #5

May 15, 2008

Description

In this assignment, you will implement the infeasible start Newton's method.

You should turn in an archive containing all of your source code, plots, and a document containing a brief description of your code (how to run it, what parameters it takes, etc), and answers to all questions. Email this archive to your TA, at cotter@tti-c.org.

Please remember to turn in a document, and to answer the questions (in particular, talk about what your plots mean)!

1 Infeasible start Newton's method

1.1 Implementation

Implement the infeasible start Newton's method (algorithm 10.2 of Boyd and Vandenberghe) for minimizing a convex objective subject to linear equality constraints:

$$\begin{aligned} \text{minimize} &: f(x) \\ \text{subject to} &: Ax = b \end{aligned}$$

Observe that the stopping criterion given in the text is:

$$(Ax = b) \wedge (\|r(x, \nu)\|_2 \leq \epsilon)$$

and that $Ax = b$ is unlikely to be *exactly* satisfied, when implemented on a computer. Instead of the above, implement the following stopping criterion:

$$(\|Ax - b\|_2 \leq \gamma) \wedge (\|r(x, \nu)\|_2 \leq \epsilon)$$

were it is understood that γ will be very small, since its purpose is simply to compensate for the lack of numerical precision in your calculations. The functions r, r_{pri} and r_{dual} are defined on page 532. Note that $Ax - b = r_{\text{pri}}(x, \nu)$, so that the above is nothing but a condition on the magnitudes of the primal and dual residuals.

Your function should take ten parameters: ∇f and $\nabla^2 f$ (the gradient and Hessian of the objective function); A and b (which define the linear constraints); x_0 and ν_0 (the initial primal and dual variables, respectively);α and β (the parameters to the backtracking line search); γ and ϵ (the stopping thresholds). Note that the backtracking line search used in algorithm 10.2 is *not* the same as the one which we have been using to this point–the line search which you will be implementing in this assignment takes both primal and dual variables into account.

Your function should return a list of 3-tuples $\left(t, r_{\text{pri}}, r_{\text{dual}}\right)$, one for every iterate, containing: the line search length t; the primal residual r_{pri}; the dual residual r_{dual}. Additionally, make sure to do the following:

- Check that $\gamma, \epsilon, \alpha, \beta > 0$, $\alpha < \frac{1}{2}$, and $\beta < 1$, raising an error if any of these conditions is not satisfied
- Check that the dimensions of the matrix A and vectors b, x_0, ν_0 are all compatible, raising an error if not
- Raise an error if either your main Newton loop *or* your line search loop performs a "ridiculous" number of iterations
- Comment any portion of your code of which the interpretation is not obvious

1.2 Experiments

In this problem, you will be reproducing one of the experiments in example 10.3.5 of Boyd and Vandenberghe, by solving the following problem:

$$\begin{aligned} \text{minimize} \quad &: \quad f(x) = -\sum_i \log x_i \\ \text{subject to} \quad &: \quad Ax = b \end{aligned}$$

Note that there is the implicit constraint above that $x \succeq 0$. The matrix A and vector b are contained in the files "a_matrix.csv" and "b_vector.csv", respectively, in the archive associated with this problem set (for which there does exist a $x \succeq 0$ such that $Ax = b$).

Run your implementation of the infeasible start Newton's method with f, A and b as above, $\gamma = 10^{-12}$, $\epsilon = 10^{-8}$, $\alpha = 0.45$ and $\beta = 5$. Create the following two plots:

- The primal residual, in log scale, versus iteration number, and likewise for the dual residual, on the same plot, with different colors and/or line styles
- The step length t versus iteration number

When does quadratic convergence begin? How do you know? What relationship do you see between the start of quadratic convergence, and the values of the residuals? Why does this happen?

In problem 1.1, it was mentioned (when discussing the stopping criterion) that $Ax = b$ will not be satisfied exactly by any iterate, due to numerical errors. Will it be satisfied *in theory* for any iterate? If so, when?