# Lab Assignment 7
# Line Search Methods for Unconstrained Optimization

ACS II

Spring 2020

Assigned: February 20, 2020

Due: February 27, 2020

## Introduction

In this lab we will look at some popular examples of line search methods for minimizing an objective function, $f(\mathbf{x})$, $\mathbf{x} = (x_1 x_2 \dots x_n)^T$. We know that $\nabla f(\mathbf{x}) = 0$ at a minimum of $f(\mathbf{x})$. In a line search method, we move toward the minimum by moving some distance, $\alpha$, along some descent direction, $\mathbf{p}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{1}$$

We will see that there are a variety of methods for choosing the descent direction $\mathbf{p}_k$, but typically (and for the methods we'll explore here), the search direction assumes the form

$$\mathbf{p}_k \ = \ -B_k^{-1} \nabla f_k \tag{2}$$

where $f_k = f(\mathbf{x}_k)$ and $B_k$ is a symmetric, nonsingular matrix. If $B_k$ is also positive definite, then we find that from the definition of positive definiteness

$$(\nabla f_k)^T \mathbf{p}_k \ = \ -(\nabla f_k)^T B_k^{-1} \nabla f_k \ < \ 0 \tag{3}$$

so $\mathbf{p}_k$ is a descent direction. Once we specify how to choose the direction and the step size, our algorithm is set. We terminate our search if we are sufficiently close to the minimum,

$$\frac{||\nabla f_k||}{(1 + |f_k|)} \leq \epsilon \tag{4}$$

where $\epsilon$ is some tolerance. Ideally, we could perform an exact line search by choosing $\alpha_k$ as the scalar that minimizes the function $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$, subject to the constraint that $\alpha > 0$.

For most problems, however, this approach is computationally too expensive, and so we use an inexact line search instead, to obtain a sufficient reduction of $f$ at a much lower cost. The Wolfe conditions put constraints on the choice of $\alpha$ to ensure that the step length is appropriate. Specifically, the Wolfe conditions include:

1. a sufficient decrease condition, to ensure that the step length leads to reduction in $f$
2. a curvature condition, to ensure that the step length is not too small.

However, if we employ *backtracking* in our algorithm for choosing $\alpha_k$, then we only need to use the first of the Wolfe conditions (the sufficient decrease condition, or the Armijo condition). The sufficient decrease condition requires that $\alpha_k$ be chosen such that

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + \gamma_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \tag{5}$$

for some constant $\gamma_1 \in (0, 1)$. The backtracking algorithm chooses a large step size and iteratively reduces the step size until the sufficient decrease condition is satisfied. Because we start with a large step size, we mitigate the danger of choosing a step size too small.

**Backtracking Line Search Algorithm**

1. Choose $\hat{\alpha} > 0$, $\rho \in (0, 1)$, $\gamma_1 \in (0, 1)$
2. Set $\alpha = \hat{\alpha}$
3. Repeat until $f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + \gamma_1 \alpha (\nabla f(\mathbf{x}_k))^T \mathbf{p}_k$:
   (a) $\alpha = \rho \alpha$
4. end repeat
5. terminate with $\alpha_k = \alpha$

For the exercises in this lab, use the backtracking line search to find the step length. (When in doubt, $\hat{\alpha} = 1$ is often a good choice.)

# Method of Steepest Descent

If we choose $B_k$ to be the identity matrix, then our search direction is the negative gradient, $\mathbf{p}_k = -\nabla f_k$. This corresponds to searching along the direction in which the function decreases most rapidly at the point $\mathbf{x}_k$.

# Newton's Method

If we apply Newton's method to find the solution to the equation $\nabla f = 0$, then we will obtain the Newton search direction! The Newton direction is given by

$$\mathbf{p}_k \;=\; -H_f(\mathbf{x}_k)^{-1}\nabla f_k \tag{6}$$

which corresponds to $B_k = H_f(\mathbf{x}_k)$ where $H_f(\mathbf{x}_k)$ is the Hessian matrix of $f_k$ (i.e. the Jacobian of $\nabla f_k$). We've already seen the higher convergence rate of Newton's method in our one-dimensional codes. However, we now must solve a linear system to find the search direction, $H_f(\mathbf{x}_k)\mathbf{p}_k = -\nabla f_k$. This method also requires us to have direct analytical knowledge of the objective function and its derivatives. Solving (6) requires us to invert $H_f(\mathbf{x}_k)$. For large problems this is expensive and dedicated linear algebra packages should be used. For the test problem (9) however $H_f(\mathbf{x}_k)$ is a 2x2 matrix and a simple analytic inverse is known. In particular we have the identity:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}. \tag{7}$$

# Broyden-Fletcher-Goldfarb-Shanno (BFGS)

Instead of using the exact Hessian (which in general may not be known, or may be expensive to compute/invert) inexact Newton methods, of which BFGS is one, use an approximation to the Hessian which is updated at each iteration.

For this method $B_{k+1}$ is defined as:

$$B_{k+1} = B_k + \alpha\mathbf{u}\mathbf{u}^T + \beta\mathbf{v}\mathbf{v}^T,$$

where

$$\mathbf{u} = \nabla f_{k+1} - \nabla f_k, \qquad \mathbf{v} = B_k\mathbf{s},$$

and

$$\mathbf{s} = \mathbf{x}_{k+1} - \mathbf{x}_k, \qquad \alpha = \frac{1}{\mathbf{u}^T\mathbf{s}}, \qquad \beta = -\frac{1}{\mathbf{s}^T\mathbf{v}}.$$

The first step of BFGS can be simply a steepest descent step, i.e. $B_0 = I$. One of the advantages of inexact Newton methods is that we do not have to directly compute the

inverse of $B_{k+1}$. Instead, using $B_k^{-1}$ we can apply the Sherman Morrison formula,

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(\mathbf{s}^T\mathbf{u} + \mathbf{u}^T B_k^{-1}\mathbf{u})(\mathbf{s}\mathbf{s}^T)}{(\mathbf{s}^T\mathbf{u})^2} - \frac{B_k^{-1}\mathbf{u}\mathbf{s}^T + \mathbf{s}\mathbf{u}^T B_k^{-1}}{\mathbf{s}^T\mathbf{u}}. \tag{8}$$

Since $B_0 = I$, we know $B_0^{-1} = I$. In addition to the number of iterations and the runtime for this method, for the first three iterations compute the matrix inverse exactly using (7) and compare it to the inverse computed using (8).

# Deliverable

We've examined three different descent methods, each with a different choice of $B_k$. Write software that implements these three methods for finding the minimum of the two-dimensional Rosenbrock function,

$$f(\mathbf{x}) \;=\; 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{9}$$

which has a global minimum at $\mathbf{x}^* = (1, 1)$. Perform a maximum of 2000 iterations, and terminate the search if the approximation reaches a tolerance of $\epsilon = 10^{-6}$. For each of the methods, demonstrate your software's results for the initial starting point $\mathbf{x}_0 = (-3, -4)$, and compare the convergence and required number of iterations for the different methods, as well as the total runtime for each method. You may use any compiled language of your choice. Please include compilation instructions with your submission.

# Submission and Grading

To get credit for this assignment, you must submit the following information to the lab instructor by 11:59pm, February 27, 2020:

- Your code.
- Your report as a single file in pdf format, including your plots, results from your work and relevant discussion of your observations, results, and conclusions.

**This information must be received by 11:59pm, February 27, 2020.** As stated in the course syllabus, late assignment submissions will be subject to a 10% point penalty per 24 hours past the due date at time of submission, to a maximum reduction of 50%, according to the formula:

*[final score] = [raw score] - min( 0.5, 0.1 * [# of days past due] ) * [maximum score]*