

## Assignment #8

**Due Monday, 19 November 2018, at the start of class (*revised*)**

Please read sections 11.3, 11.4, 11.5, 12.2, 12.3 of the textbook.

DO THE FOLLOWING §11.5 EXERCISE FROM PAGE 386:

- Exercise 5.5. (*Note this is for any differentiable function. Please assume that the line search is exact. You do not need to address how it would be implemented.*)

DO THE FOLLOWING §12.2 EXERCISE FROM PAGE 408:

- Exercise 2.3. (*Make sure to calculate  $Q = \nabla^2 f(x)$  and then  $\text{cond}(Q)$ . For the last “What conclusions . . .” question, answer: What is the rate of convergence of  $f(x_k)$ ?*)

DO THE FOLLOWING §12.3 EXERCISE FROM PAGE 521:

- Exercise 3.7. (*Note that  $B_k s_k = y_k$  is the “secant condition” on pages 412–413.*)

**Problem P13.** Let  $\phi$  be any number greater than one. Suppose the back-tracking line search algorithm chooses  $\alpha_k$  as the first element of the sequence

$$1, 1/\phi, 1/\phi^2, 1/\phi^3, \dots$$

satisfying the usual sufficient decrease condition with  $0 < \mu < 1$ , namely

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \mu \alpha_k p_k^\top \nabla f(x_k).$$

What value of  $\phi$  does the proof of Theorem 11.7 use? Describe how to modify that proof so that the conclusion still holds with any  $\phi > 1$ . (*Suggestion: Quote from the proof those parts that need to change. Then state the new version of those parts.*)

**Problem P14.** Consider the one-variable problem

$$\min_{x \in \mathbb{R}} f(x)$$

where  $f : \mathbb{R} \rightarrow \mathbb{R}$  is twice continuously-differentiable. Recall that the Newton method to solve  $f'(x) = 0$ , that is, the Newton method for the above minimization problem, is given by the formulas  $p_k = -f''(x_k)^{-1} f'(x_k)$  and  $x_{k+1} = x_k + p_k$ .

The *secant method* for minimization only differs from the Newton method by replacing the second derivative with a difference quotient approximation based on the last two iterates:

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}.$$

Thus the secant method computes the step (search vector) by

$$p_k = -\frac{(x_k - x_{k-1})f'(x_k)}{f'(x_k) - f'(x_{k-1})}$$

- (a) Implement the secant method.
- (b) Use your code to solve the following problems and initial iterates:
  - i)  $f(x) = 3x^4 - 4x^3 + 3x^2 - 6x$ ,  $x_0 = -1$ ,  $x_1 = 0$
  - ii)  $f(x) = x^2 - 2 \sin x$ ,  $x_0 = 0$ ,  $x_1 = 1$
- (c) In (b) i) the exact minimum is at  $x_* = 1$ . Compute the errors  $e_k = x_k - x_*$ . Give evidence that the convergence is superlinear. Using the notation of section 2.5, what is your estimate of the exponent  $r$ ?
- (d) Describe the performance of the secant method when  $f(x)$  is a quadratic function with a unique minimum.

**Problem P15.** Consider the problem

$$\min_{x \in \mathbb{R}^n} f(x).$$

- (a) Implement the symmetric rank-one method, described on page 414, to solve this problem. That is, write a code

```
function [xk, xklist] = srlbt(x0, f, tol)
```

which is a quasi-Newton method which

- i) initially sets  $B_0 = I$ , so the first step is steepest-descent,
- ii) uses the symmetric rank-one formula (page 414) to update  $B_k$  to  $B_{k+1}$ ,
- iii) reverts to a steepest-descent step if a step is not a descent direction, and
- iv) uses back-tracking line search.

The input  $x_0$  is a length  $n$  vector for the initial iterate, then input  $f$  is a function which returns both the function value and the gradient,

```
function [fx, dfx] = f(x)
```

and the input  $tol$  is for the stopping criterion  $\|\nabla f(x_k)\| \leq tol$ . Base your code on the steepest-descent-with-back-tracking code already written:

[bueler.github.io/M661F18/matlab/sdbt.m](https://bueler.github.io/M661F18/matlab/sdbt.m)

This code shows how to handle the inputs and outputs. It shows how to return the final iterate  $x_k$  and a list of iterates  $xklist$ .

- (b) Test your code on the 2D quadratic function

$$f_{2D}(x) = 5x_1^2 + \frac{1}{2}x_2^2,$$

using initial iterate  $x^{(0)} = (1, 1)^\top$ , and on the 5D quadratic function

$$f_{5D}(x) = 10x_1^2 + 5x_2^2 + x_3^2 + \frac{1}{2}x_4^2 + \frac{1}{10}x_5^2,$$

using initial iterate  $x^{(0)} = (1, 1, 1, 1, 1)^\top$ . Use  $tol = 10^{-6}$ . How many iterations are needed in each case?

- (c) If you implemented the Newton method with back-tracking, how would its performance compare on these functions? (*No implementation needed.*)
- (d) On each of the functions in (b), compare actual performance, namely number of iterations, to steepest-descent-with-backtracking, i.e. using `sdbt` above. Again use  $tol = 10^{-6}$ .