

CS726, Fall 2008  
Homework 3  
(due Fri 9/26/08)

Submit the two Matlab code files electronically using the instructions on the course web page. Once you have set up the path in your user directory, you need to put your files for submission in a single directory and run the command

```
handin -c cs726-1 -a hwk3 -d <directory name>
```

Electronically hand in exactly two files with the names `SteepDescent.m` and `StepSize.m`. Hand in a hard copy of your answer to the first two questions in class.

1. Exercise 3.2 from the text. Construct a counterexample of a real function  $\phi$  for which no step lengths  $\alpha$  satisfy the Wolfe conditions for  $c_1, c_2$  satisfying the conditions of the question.
2. Exercise 3.5 from the text.
3. Write a Matlab routine called `StepSize.m` to implement Algorithm 3.5 in the text, which includes the “zoom” procedure of Algorithm 3.6.
  - Set  $\alpha_{\max}$  to be the Matlab-defined quantity “realmax”.
  - In the extrapolation step of Algorithm 3.5 (where you choose  $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$ ), simply set  $\alpha_{i+1} = 3\alpha_i$ .
  - In the interpolation step of the zoom procedure, perform quadratic interpolation using the values of  $\phi$  and  $\phi'$  at  $\alpha_{lo}$  and the value of  $\phi$  at  $\alpha_{hi}$  (you can modify formula (3.58) appropriately). Use safeguarding to ensure that  $\alpha_j$  is not within a distance  $0.2|\alpha_{hi} - \alpha_{lo}|$  of the endpoints.

The header line of your routine should be

```
function [alfa,x] = StepSize(fun, x, d, alfa, params)
```

where the input parameters are:

**fun** - a pointer to a function (such as `obja`, `objb`, `objc`)

**x** - a structure with three fields `x.p`, `x.f`, and `x.g` in which `x.p` contains the point  $x$ , while `x.f` and `x.g` contain the function and gradient values corresponding to  $x$ . On input, `x.p` is set to the starting point values `x = struct('p', [-1.2, 1.0])`; while `x.f` and `x.g` are set to the corresponding function and gradient values.

**d** - a vector containing the search direction

**alfa** - the initial value of the step length (set it to 1 for this project)

**params** - a structure containing parameter values for the test:

```
params struct('c1',0.01,'c2',0.5,'maxit',100);
```

The routine should call on **fun** to evaluate the objective function and gradient as computed points, using

```
x.f = feval(fun,x.p,1);
```

and

```
x.g = feval(fun,x.p,2);
```

respectively.

The output parameters are:

**alfa** - the value  $\alpha_k$  satisfying the strong Wolfe conditions

**x** - on output, **x.p** contains the final vector  $x_{k+1} = x_k + \alpha_k d_k$  while **x.f** and **x.g** contain the function and gradient values at  $x_{k+1}$ .

The purpose of returning these values is to avoid computing them again before the next call to the StepSize routine.

4. Test your routine by writing a Matlab program **SteepestDescent.m** to implement the steepest descent method, with  $d_k = -\nabla f(x_k)$ . Terminate when either  $\|\nabla f(x_k)\|_2 \leq 10^{-4}$  or 5000 function evaluations have been taken, whichever comes first. The first line of the function should be

```
function [inform,x] = SteepDescent(fun,x,sdparams)
```

The inputs **fun** and **x** are as described above, while **sdparams** is the following structure:

```
sdparams = struct('maxit',10000,'toler',1.0e-4);
```

The output **inform** is a structure containing two fields: **inform.status** is 1 if the gradient tolerance is achieved and 0 if not, while **inform.iter** is the number of steps taken. The output **x** is the solution structure, with point, function, and gradient values at the final value of  $x_k$ .

5. Matlab functions that implement the three functions below can be found in the public directory of the course web site and the class volume, under the names **obja.m**, **objb.m**, and **objc.m**. Your program will be tested using the code **hwk3.m** on the class web site. For each of the three functions above, the code will be run with initial point  $x_0 = (-1.2, 1)^T$ .

(a)  $f(x) = x_1^2 + 5x_2^2 + x_1 - 5x_2$

(b)  $f(x) = x_1^2 + 5x_1x_2 + 100x_2^2 - x_1 + 4x_2$

(c)  $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

Note that the global variables `numf` and `numg` are reported by the program `hwk3.m` and incremented by the function evaluation routines. Be sure to set these to zero at the start of `SteepDescent.m`.

Do not print out the value of  $x$  at each iteration!

Hint: Your code for `StepSize.m` could contain two sections. The first section implements Algorithm 3.5, whose main purpose is to identify a bracket within which an acceptable value of  $\alpha_k$  lies. When the conditions for a call to `zoom` are satisfied, you could assign the values of  $\alpha_{lo}$  and  $\alpha_{hi}$ , and jump out of the main “repeat” loop using a Matlab `break` statement. This will take you to the second section of code, which implements the `zoom` procedure.