```
 1 function [x_opt, fval_opt, x_iter, f_iter, alpha] = SteepestDescent(x0, f, G,↙
maxiter, grad_tol)
 2 % Inputs
 3 % x0: initial point
 4 % f: function handle
 5 % G: function handle for the gradient of f
 6 % maxiter: maximum number of iterations (default 10000)
 7 % grad_tol: minimum norm value of gradient (default 1e-4)
 8 %
 9 % Returns
10 % x_opt: x^*
11 % fval_opt: f(x^*)
12 % x_iter: all iterates k -- column k contains x_k
13 % f_iter: a vector of all function values f(x_k)
14 % alpha: a vector of all step lenghts alpha_k
15
16 % Termination criteria
17 maxiter_default  = 10000;
18 grad_tol_default = 1e-4;
19
20 if nargin == 3
21     maxiter = maxiter_default;
22     grad_tol = grad_tol_default;
23 elseif nargin == 4
24     grad_tol = grad_tol_default;
25 end
26
27 nx = size(x0,1); % Number of variables
28
29 % Declare some variables
30 x     = NaN(nx,maxiter);
31 p     = NaN(nx,maxiter);
32 grad  = NaN(nx,maxiter);
33 alpha = NaN(1,maxiter);
34 fval  = NaN(1,maxiter);
35
36
37 % Do some calcualtions before the while loop. I.e., do the first iteration:
38 k = 1; % iteration number
39 x(:,k) = x0;
40 % look at the different functions below and finish them before continuing.
41 fval(k) = f(x0);
42 grad(:,k) = G(x0);
43 p(:,k) = sd(grad(:,k));
44
45 alpha_0 = 1;
46
47 alpha(k) = linesearch(f, x(:,k), p(:,k), fval(k), grad(:,k), alpha_0);
48 x(:,k+1) = x(:,k) + alpha(k) * p(:,k);
49 grad(:,k+1) = G(x(:,k+1));
50 k = k + 1;
51
52 while k < maxiter && norm(grad(:,k),2) >= grad_tol
53     fval(k) = f(x(:,k));    % Evaluate the Rosenbrock function
54     p(:,k) = sd(grad(:,k)); % Calculate steepest-descent direction based on↙
gradient
55     alpha_0 = alpha(k-1) * ((grad(:,k-1).')*p(:,k-1))/((grad(:,k).')*p(:,k)); % N%W↙
p. 59
56     alpha(k) = linesearch(f, x(:,k), p(:,k), fval(k), grad(:,k), alpha_0); %↙
Determine alpha using Alg. 3.1
```

```
57      x(:,k+1) = x(:,k) + alpha(k) * p(:,k);
58      grad(:,k+1) = G(x(:,k+1));
59      k = k + 1;
60  end
61  fval(k) = f(x(:,k)); % Final function value
62
63  % Delete unused space
64  x = x(:,1:k);
65  p = p(:,1:k);
66  grad = grad(:,1:k);
67  alpha = alpha(1:k);
68  fval = fval(1:k);
69
70  % Return values
71  x_opt = x(:,end);
72  fval_opt = f(x_opt);
73  x_iter = x;
74  f_iter = fval;
75
76  end
77
78  % Function returning the steepest-descent direction based on the gradient
79  % of f
80  function p = sd(grad)
81      p = -grad;
82  end
83
84  % Function implementing Algorithm 3.1, page 37 in N&W
85  function alpha_k = linesearch(f, xk, pk, fk, gradk, alpha_0)
86      alpha = alpha_0;
87      rho = 0.9;
88      c1 = 0.5; % a constant for sufficient decrease
89      while f(xk + alpha*pk) > fk + c1*alpha*(gradk.')*pk
90          alpha = rho * alpha;
91      end
92      alpha_k = alpha;
93  end
94
```