The important question is therefore how to choose the constant stepsize so that it will not be too large (to ensure convergence) and not too small (to ensure that the convergence will not be too slow). We will consider again the theoretical issue of choosing the constant stepsize in Section 4.7. ∎

Let us now consider an example with a backtracking stepsize selection rule.

Example 4.9 (stepsize selection by backtracking). The following MATLAB function implements the gradient method with a backtracking stepsize selection rule.

```
function [x,fun_val]=gradient_method_backtracking(f,g,x0,s,alpha,...
  beta,epsilon)
% Gradient method with backtracking stepsize rule
%
% INPUT
%======================================
% f ......... objective function
% g ......... gradient of the objective function
% x0......... initial point
% s ......... initial choice of stepsize
% alpha ..... tolerance parameter for the stepsize selection
% beta ...... the constant in which the stepsize is multiplied
%             at each backtracking step (0<beta<1)
% epsilon ... tolerance parameter for stopping rule
% OUTPUT
%======================================
% x ......... optimal solution (up to a tolerance)
%             of min f(x)
% fun_val ... optimal function value
x=x0;
grad=g(x);
fun_val=f(x);
iter=0;
while (norm(grad)>epsilon)
    iter=iter+1;
    t=s;
    while (fun_val-f(x-t*grad)<alpha*t*norm(grad)^2)
        t=beta*t;
    end
    x=x-t*grad;
    fun_val=f(x);
    grad=g(x);
    fprintf('iter_number = %3d norm_grad = %2.6f fun_val = %2.6f \n',...
    iter,norm(grad),fun_val);
end
```

As in the previous examples, we will consider the problem of minimizing the function x^2+2y^2. Employing the gradient method with backtracking stepsize selection rule, a starting vector $\mathbf{x}_0=(2,1)^T$ and parameters $\varepsilon=10^{-5}, s=2, \alpha=\frac{1}{4}, \beta=\frac{1}{2}$ results in the following output:

```
>> A=[1,0;0,2];
>> [x,fun_val]=gradient_method_backtracking(@(x)x'*A*x,@(x)2*A*x,...
[2;1],2,0.25,0.5,1e-5);
iter_number =   1 norm_grad = 2.000000 fun_val = 1.000000
iter_number =   2 norm_grad = 0.000000 fun_val = 0.000000
```

That is, the gradient method with backtracking terminated after only 2 iterations. In fact, in this case (and this is probably a result of pure luck), it converged to the *exact*