Contents


```
function info_matrix = backtrack_line_search(method,f,x_initial,param)
```

H1_BACKTRACK_LINE_SEARCH Introduction

```
         Returns the first, second, and last points found for the
         minimization process.

method: The method of optimization used, given as a string, with
        the options of "steepd" for steepest descent method, and
        "newton" for Newton's method.

f:      The function to be evaluated, using symbolic variables

x_0:    The initial point to begin the minimization search, given
        as a vertical vector.

param:  The basic parameters, given in a vector with the format:
        [alpha, rho, c, tolerance]
```

Establishing Parameters

```
    info_matrix = zeros(1,6);
    format long

    a = param(1);
    r = param(2);
    c = param(3);
    x_0 = x_initial;
    tolerance = param(4);
```

Setting Up Functions

```
    x                   = sym('x', [2,1]);
    sfun_f(x)           = f;
    sfun_f_gradient(x)  = gradient(sfun_f);
    sfun_f_hessian(x)   = hessian(sfun_f);

    f_eval              = matlabFunction(sfun_f, 'Vars', {x});
    f_grad              = matlabFunction(sfun_f_gradient, 'Vars', {x});
    f_hess              = matlabFunction(sfun_f_hessian, 'Vars', {x});
```

Backtracking Line Search Algorithm

```
    i    = 1;
    while abs(f_eval(x_0)) > tolerance && norm(f_grad(x_0)) > tolerance

        if method == "steepest descent"
            p_k = -f_grad(x_0) / norm(f_grad(x_0));
            p_k = p_k ./ norm(p_k);
        elseif method == "Newton"
            p_k = -inv(f_hess(x_0))' * f_grad(x_0);
        end

        while f_eval(x_0 + a*p_k) > f_eval(x_0) + c*a*p_k'*f_grad(x_0)
            a = r * a;
        end

        info_matrix(i,:) = [x_0', f_eval(x_0), p_k', a];
        i = i + 1;

        x_0 = x_0 + a * p_k;

    end

    info_matrix(i,:) = [x_0', f_eval(x_0), NaN, NaN, NaN];
```

Printing Results

```
    len = length(info_matrix(:,1));
    disp(len-1 + " iterations using " + method + " method,")
    disp("starting at point (" + x_initial(1) + ", " + x_initial(2) + "):")

    headers = {'x_1','x_2','f(x_0)','p_k1','p_k2','alpha'};
    s = '...';  space = {s, s, s, s, s, s};
    firsts = num2cell(info_matrix(1:10,:));
    last = num2cell(info_matrix(end-1:end,:));
    disp([headers; firsts; space; last]);
    fprintf('\n')

end
```

Published with MATLAB® R2019a