```
function [x_1, x_2] = Newtons_Method(x_exact, x_k, fx, dfx, max_iter, error_tol)

% Use Newton's Method to find the two solutions that are nearest to a solution.
% x_exact   - The exact solution
% x         - A set of x
% fx        - Function of x
% dfx       - First Derivative of fx
% max_iter  - Maximum Interation to exit the function
% error_tol - Acceptable precision of the solution

% Initialize x_vector that stores the final x_k(n) solutions
x_vector = zeros(length(x_k), 1);

for iter = 1 : length(x_k)

    % Initialize x_0
    x_0 = x_k(iter);

    % Initialize x_n
    x_n = [];

    % Calculate x_1
    x_n(1) = x_0 - fx(x_0) / dfx(x_0);

    % Start with counter 2 since x_o and x_1 were calculated
    counter = 2;

    % Apply Newton's method
    while ((abs(x_n(counter-1) - x_exact) > error_tol) && (counter <= max_iter))

        x_n(counter) = x_n(counter-1) - fx(x_n(counter-1)) / dfx(x_n(counter-1));

        counter = counter + 1;

    end

    x_vector(iter) = x_n(counter-1);

end

% Determine the difference between calculated x and exact x
difference_x = abs(x_vector - x_exact);

% Determine the minimum x
[~, index_1] = min(difference_x);

% Define array size
sorted_difference_x = unique(difference_x);

% Find the second closest solution.
for iter = 1: length(difference_x)

    if difference_x(iter) == sorted_difference_x(2)

        index_2 = iter;

        break

    end

end
```

```
% Assign to x_1
x_1 = x_vector(index_1);

% Assign to x_2
x_2 = x_vector(index_2);


end
```