**Algorithm 2: Newton method**

|  |
| --- |
| Method introduction: |
| he idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the x-intercept of this tangent line (which is easily done with elementary algebra). This x-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated  Shortcomings:   1. Affected by the starting point; 2. It is difficult to calculate the derivative of partial functions |
| Algorithm Design |
| x0 = 1 %The initial value  f = @(x) x^2 - 2 %The function whose root we are trying to find  fprime = @(x) 2\*x %The derivative of f(x)  tolerance = 10^(-7) %7 digit accuracy is desired  epsilon = 10^(-14) %Don't want to divide by a number smaller than this  maxIterations = 20 %Don't allow the iterations to continue indefinitely  haveWeFoundSolution = false %Have not converged to a solution yet  for i = 1 : maxIterations  y = f(x0)  yprime = fprime(x0)  if(abs(yprime) < epsilon) %Don't want to divide by too small of a number  % denominator is too small  break; %Leave the loop  end  x1 = x0 - y/yprime %Do Newton's computation  if(abs(x1 - x0) <= tolerance \* abs(x1))  haveWeFoundSolution = true  break; %Done, so leave the loop  end  x0 = x1 %Update x0 to start the process again  end  if (haveWeFoundSolution)  ... % x1 is a solution within tolerance and maximum number of iterations  else  ... % did not converge  end |
| Matlab code |
| function [x, y] = MyNewton(fun, dfun, x1, tol, max)  % This is the code for Newton's method.  % Input:  % x1 Initial guess  % fun function  % dfun derivative of the function  % tol Allowable tolerance in computed zero  % max Maximum number of iterations  % Output:  % x Vector of approximations to zero  % y Vector of function values, fun(x)  % Preallocate vectors.  x = zeros(max, 1);  y = zeros(max, 1);  dy = zeros(max, 1);  % Set an intial interval.  x(1) = x1;  y(1) = feval(fun, x(1));  dy(1) = feval(dfun, x(1));  % Newton search  for i = 2 : max  x(i) = x(i-1) - y(i-1)/dy(i-1);  y(i) = feval(fun, x(i));  if y(i) == 0  fprintf('Exact solution found\n');  break;  end  if (abs(x(i) - x(i-1)) < tol)  fprintf('Newton method has converged\n');  break;  end  dy(i) = feval(dfun, x(i));  iter = i+1;  end  if (iter > max)  fprintf('Zero not found to desired tolerance within the maximum number of iterations\n');  end  % Output results  k = 1:iter;  fprintf(' iter x y\n');  disp([k' x(1:iter) y(1:iter)]); |
| Examples and Result |
| Remarks |
| 此处写该方法程序设计的一些注意事项，也可以空白 |
|  |