The Solution of Single Nonlinear Equations $f(x)=0$: Newton's Method and Beyond

BY WHZECOMJM ZHANG

South University of Science and Technology of China
10th Mar 2013

Abstract

We first explore several elementary methods of finding solution of sigle nonlinear equations, and give Matlab code of such methods and give some new methods beyond Newton's method. What's more, we discussed the applicability of Newton's method and give and prove an counter example of converge of Newton's Method.

Table of contents

1 Solution of A Single Nonlinear Equation

One of the most frequently occurring problems in scientific work is to find the roots of equations of the form

$$f(x)=0 \tag{1}$$

The function $f(x)$ may be given explicitly as, for example, a polynomial or a transcendental function. Frequently, however, $f(x)$ may be known only implicitly in that only a rule for evaluating it on any argument is known. In rare cases it may be possible to obtain the exact roots of equation (1) such as in the case of a factorizable polynomial. In general, however, we can hope to obtain only approximate solutions, relying on some computational techniques to produce the approximation.the algorithms for any of the methods presented herein will not provide the exact solution to the equation (1), instead, we will stop the algorithm when the equation is satisfied within an allowed tolerance or error, ε . In mathematical terms this is expressed as [1]

$$|f(x_R)|<\varepsilon \tag{2}$$

The value of x for which the nonlinear equation (1) is satisfied, i.e., $x=x_R$, will be the solution, or root, to the equation within an error of ε units.

In this article, we will first introduce some elementary iterative methods for finding a root of equation (1).

2 The Bisection Method

2.1 Introduction of Bisection Method

We start with an initial interval $[a,b]$, where $f(a)$ and$f(b)$ have opposite signs. Since the graph$y=f(x)$ of a continuous function is unbroken, it will cross the x-axis at a zero $x=r$ that lies somewhere in the interval. The bisection method systematically moves the end points of the interval closer and closer together until we obtain an interval of arbitrarily small width that brackets the zero. The decision step for this process of interval halving is first to choose the midpoint $c=(a+b)/2$ and then to analyze the three possibilities that might arise:

- If $f(a)$ and$f(c)$ have opposite signs, a zero lies in $[a,c]$;
- If $f(c)$ and$f(b)$ have opposite signs, a zero lies in $[c,b]$;
- If $f(c)=0$; then the zero is c.

If either case 1 or 2 occurs, we have found an interval half as wide as the original interval that contains the root, and we are squeezing down on it. To continue the process, relabel the new smaller interval $[a,b]$ and repeat the process until the interval is as small as desired.

2.2 Matlab Code of Bisection Method

Now we give the Matlab code of bisection method through the pesudo code first:

```
for $i=0,1,2,\ldots$, until satisfied, do
    $m \leftarrow (a_i+b_i)/2$
    if $f(a_i)f(m)\leq 0$
        then $a_{i+1} \leftarrow a_i$
             $b_{i+1} \leftarrow m$
        else $a_{i+1} \leftarrow m$
```

$b_{i+1} \leftarrow b_i$

After giving pesudo code, we can easily give the matlab code of bisection method as follows:

```
function root=BisectionRoot(f,a,b,eps)
if(nargin==3)
   eps=1.0e-5;
end
f1=subs(sym(f),findsym(sym(f)),a);
f2=subs(sym(f),findsym(sym(f)),b);
if(f1==0)
   root=a;
end
if(f2==0)
   root=b;
end
format long;
if(f1*f2>0)
   disp('two end points have same symbol')
return
else
   root=FindRoots(f,a,b,eps);
end

function r=FindRoots(f,a,b,eps)
   f_1=subs(sym(f),findsym(sym(f)),a);
   f_2=subs(sym(f),findsym(sym(f)),b);
   mf=subs(sym(f),findsym(sym(f)),(a+b)/2);
   if(f_1*mf>0) t=(a+b)/2;
      r=FindRoots(f,t,b,eps);
   else if(f_1*mf==0)
      r=(a+b)/2;
    else if(abs(a-b)<=eps)
      r=(b+3*a)/4;
     else s=(a+b)/2;
   r=FindRoots(f,a,s,eps);
     end
    end
   end
```

2.3 Application of Bisection Method Through Code

Example 1. $f(x) = x^3 - 3x + 1$

Since $f(x)$ is a cubic it has either one or three real zeros. For $f(-2) = -1$, $f(-1) = 3$, $f(1) = -1$, $f(2) = 3$, Thus $f(x)$ must have three zeros instead of one. These zeros are initially bracketed by $[-2, -1]$, $[-1, 1]$, $[1, 2]$. The iteration results are given in the following(defalt precision is 1.0e-5):

- [-2,-1] case: -1.879384994506836

a	b	$\frac{a+b}{2}$	$f(a)$	$f\left(\frac{a+b}{2}\right)$	$f(b)$
-2	-1	-1.5	-1	2.125	3

- [-1,1] case: 0.347296237945557

a	b	$\frac{a+b}{2}$	$f(a)$	$f\left(\frac{a+b}{2}\right)$	$f(b)$
-1	1	0	3	1	-1

- [1,2] case: 1.532088875770569

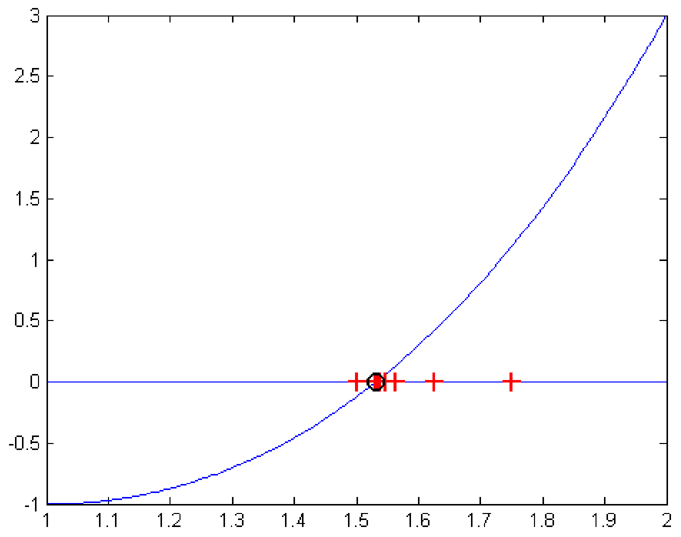
a	b	$\frac{a+b}{2}$	$f(a)$	$f\left(\frac{a+b}{2}\right)$	$f(b)$
1	2	1.5	-1	-0.125	3

2.4 Improved Code with animation

There is another improved code with animation to let us observe more precisely, here is the code:

```
function [xc,nost] = mybisect(f,a,b,tol)%% bisection method
  if(nargin==3)
    tol=1.0e-4;
  end
fplot(f,[a b]);
hold on;
fplot('0',[a b]);
hold on;
format long;
fa=subs(sym(f),findsym(sym(f)),a);
fb = subs(sym(f),findsym(sym(f)),b);
if (fa*fb >= 0)
   disp('f(a)f(b)<0 not satisfied!');
  return;
end
nost = 0; %number of bisection steps
while (b-a)/2 > tol
   c = (a+b)/2; % compute midpoint of intervall
   nost = nost+1; % count number of bisection steps
   plot(c,0,'r+','MarkerSize',10,'LineWidth',2)
   drawnow
   hold on;
   pause(0.5) % wait for 0.5 seconds to obtain nice movie
   fc = subs(sym(f),findsym(sym(f)),c); % evaluate f at c
     if fc == 0 % c is a solution, done
       break
     end
     if sign(fc)*sign(fa) < 0
     b = c;
     fb = fc;
     else %
     a = c;
     fa = fc;
     end
   end
xc = (a+b)/2; % new midpoint is best estimate
plot(xc,0,'ko','MarkerSize',10,'LineWidth',2)
drawnow
```

We look back to our Example 1, take the [1,2] case, we can have the following picture:



3 Secant Method

3.1 Introduction of Secant Method

Another very popular modification is the secant method. It retains the use of secants throughout, but gives up the bracketing of the root. The method starts with two estimates x_0 and x_1 and iterates as follows: [2]

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \tag{3}$$

3.2 Matlab Code of Secant Method

Now we give the Matlab code of secant method:

```
function root=SecantRoot(f,a,b,eps)
  if(nargin==3)
     eps=1.0e-6;
  end
format long;
f1=subs(sym(f),findsym(sym(f)),a);
f2=subs(sym(f),findsym(sym(f)),b);
  if(f1==0)
     root=a;
  end
  if(f2==0)
     root=b;
  end
  if(f1*f2>0)
     disp('values of end points are same symbol')
  return
  else
    tol=1;
    fa=subs(sym(f),findsym(sym(f)),a);
    fb=subs(sym(f),findsym(sym(f)),b);
```

```
      root=a-(b-a)*fa/(fb-fa);
         while(tol>eps)
            r1=root;
            fx=subs(sym(f),findsym(sym(f)),r1);
            s=fx*fa;
              if(s==0)
                 root=r1;
              else if(s>0)
                 root=b-(r1-b)*fb/(fx-fb);
              else root=a-(r1-a)*fa/(fx-fa);
              end
          end
      tol=abs(root-r1);
     end
    end
```

3.3 Application of Secant Method Through Code

We consider the same example, and what's more, let us research the interval $[1,2]$ only, then we get the root is 1.532086406410894, however, it's worse than the bisection method in a way.

4 Newton's Method

4.1 Introduction of Newton's Method

In the secant method, we can write

$$x_{i+1} = x_i - \frac{f(x_i)}{F[x_i, x_{i-1}]} \tag{4}$$

where $F[x_i, x_{i-1}] = (f(x_i) - f(x_{i-1}))/(x_i - x_{i-1})$ is a first order divided difference that approximates the first derivative of f. [3] Now, in the continuous case, the above suggests the formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{5}$$

4.2 Matlab Code for Newton's Method

Now we give the Matlab code of Newton's method:

```
function root=NewtonRoot(f,a,eps)
if(nargin==2)
   eps=1.0e-6;
end
format long;
f1=subs(sym(f),findsym(sym(f)),a);
if(f1==0)
   root=a;
else
  tol=1;
  fun=diff(sym(f));
  fa=subs(sym(f),findsym(sym(f)),a);
  dfa=subs(sym(fun),findsym(sym(fun)),a);
  root=a-fa/dfa;
    while(tol>eps)
       r1=root;
```

```
            fx=subs(sym(f),findsym(sym(f)),r1);
            dfx=subs(sym(fun),findsym(sym(fun)),r1);
            root=r1-fx/dfx;
            tol=abs(root-r1);
        end
    end
```

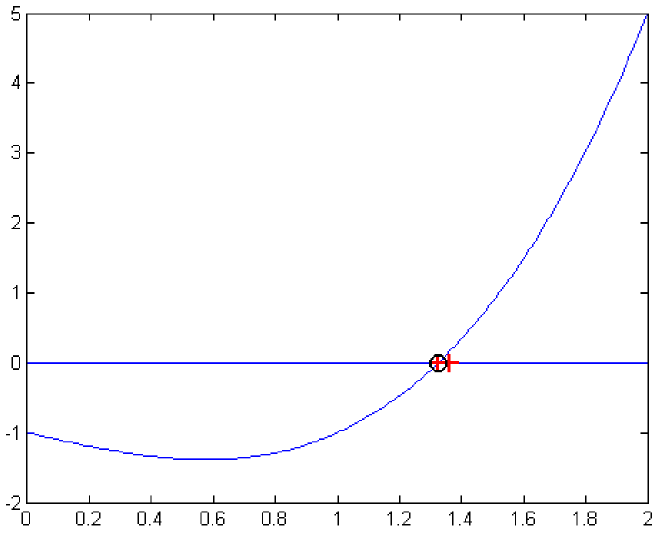
4.3 Application of Newton's Method Through Code

Example 2. We now run Newton's method to find the unique real root of $f(x)=x^3-x-1$ using $f'(x)=3x^2-1$.

We start from $x_0=a=0$, we can get root is 1.324717957245390 with precision is 1.0e-6.

a	$f(a)$	$f'(a)$	$a-\frac{f(a)}{f'(a)}$
0	-1	-1	-1

More beautifully, we can get a similar animation with bisection method, and we will get following picture:



4.4 Applicability of Newton's Method

Theorem 3. *Let f be a twice continuously differentiable function over $[a,b]$. Suppose f satisfies the following conditions:*

i. $f(a)f(b)<0$;

ii. $f'(x)\neq 0$ *for all* $x\in[a,b]$;

iii. $f''(x)$ *is either non-negative everywhere on* $[a,b]$ *or non-positive everywhere on* $[a,b]$;

iv. $\left|\frac{f(a)}{f'(a)}\right|<b-a$ *and* $\left|\frac{f(b)}{f'(b)}\right|<b-a$.

Then Newton's method converges to the unique root of $f(x)$in $[a,b]$ for any initial point $x_0\in[a,b]$.

We will not prove this theorem, on the countrary, we prove the following counter example:

Example 4. $f(x)=\arctan(x)$, as we known, $f(0)=0$, and the Newton's iteration is defined by

$$x_{k+1}=x_k-(1+x_k^2)\arctan(x) \tag{6}$$

If we choose x_0 such that

$$\arctan|x_0| > \frac{2|x_0|}{1+x_0^2} \tag{7}$$

then the sequence $|x_k|$ diverges, that is, $\lim_{k\to\infty}|x_k| = \infty$.

To see this, consider the function $g(x) := x - (1+x^2)\arctan x$, then we have $g'(x) - 2x\arctan x < 0, \forall x \neq 0$. Then we have $x_{k+1}x_k = g(x_k)x_k < 0$, and $\frac{|x_{k+1}|}{|x_k|} > \frac{|x_k|}{|x_{k-1}|}$. Finally, we have

$$|x_k| = \frac{|x_k|}{|x_{k-1}|}\frac{|x_{k-1}|}{|x_{k-2}|}\cdots\frac{|x_1|}{|x_0|}|x_0| > \left(\frac{|x_1|}{|x_0|}\right)^k|x_0| \tag{8}$$

Thus the sequence diverges.

5 Homotopy Perturbation Technique

5.1 Introduction of Homotopy Perturbation

Consider the nonlinear equation (1), we assume that α is a simple root of (1) and γ is an initial guess sufficiently close to α. We can rewrite the nonlinear equation (1) as a coupled system using the Taylor's series and technique of He [5] as

$$f(\gamma) + f'(\gamma)(x-\gamma) + g(x) = 0 \tag{9}$$

$$g(x) = f(x) - f(\gamma) + f'(\gamma)(x-\gamma) \tag{10}$$

We can rewrite form as

$$x = \gamma - \frac{f(\gamma)}{f'(\gamma)} - \frac{g(x)}{f'(\gamma)} \tag{11}$$

It's easy to see that by equations (9)&(10)

$$f(x_0) = g(x_0) \tag{12}$$

Now we rewrite equation (11) in the following equivalent useful form

$$x = c + N(x) \tag{13}$$

where $c = \gamma - \frac{f(\gamma)}{f'(\gamma)}$ and $N(x) = -\frac{g(x)}{f'(\gamma)}$.Here $N(x)$ is a nonlinear operator. We use the homotopy perturbation technique to develop a class of new iterative methods for solving nonlinear equation (1). Following the homotopy perturbation technique, one usually defines $H(\nu,p): R\times[0,1]\times R \to R$ as

$$H(\nu,p) = (1-ph)[L(\nu) - L(x)] + ph[L(\nu) + N(\nu) - c] = 0 \tag{14}$$

or

$$H(\nu,p) = L(\nu) - L(x_0) + ph[L(x_0) + N(\nu) - c] = 0 \tag{15}$$

where $p \in [0,1]$ is an embedding parameter, $h \neq 0$ is an auxiliary parameter, and x_0 is an initial approximation.Clearly from (14)&(15):

$$H(\nu,0,T) = L(\nu) - L(x_0) = x - c = 0 \tag{16}$$

$$H(\nu,1,T) = h\,[L(\nu) + N(\nu) - c] = h[\nu - c - N(\nu)] = 0 \tag{17}$$

The embedding parameterpincreases monotonically from zero to unity as trivial problem $H(\nu, 0, T) = x - c = 0$is continuously deformed to original problem $H(\nu,1,T) = h[\nu - c - N(\nu)] = 0$. The changing process of p from zero to unity is called deformation. $L(\nu) - L(x_0)$ and $L(\nu) + N(\nu) - c$ are homotopic.

we can rewrite (13) as follows by expanding $N(x)$ into a Taylor series around x_0 as

$$\nu - c - ph\left\{N(x_0) + (x-x_0)N'(x_0) + \frac{(x-x_0)^2}{2}N''(x_0) + \ldots\right\} = 0 \tag{18}$$

The basis assumption is that the solution of equations (14) and (15) can be expressed as a power series in p

$$\nu = \nu_0 + p\nu_1 + p^2\nu_2 + ... \tag{19}$$

The approximate solution of (13) can be readily obtained as

$$x = \lim_{p\to 1} \nu = x_0 + x_1 + x_2 + ... \tag{20}$$

Substitude (19) into (18), then we will get

$$x_0 + px_1 + px_2^2 + ... - x - ph\left\{ N(x_0) + (x - x_0)N'(x_0) + \frac{(x - x_0)^2}{2} N''(x_0) + ... \right\} = 0 \tag{21}$$

Equating the coefficients of the identical powers of p, we obtain

$$\begin{aligned} p^0 &: \ x_0 - c = 0 & (22)\\ p^1 &: \ x_1 - hN(x_0) = 0 & (23)\\ p^2 &: \ x_2 - hx_1N'(x_0) = 0 & (24)\\ p^3 &: \ x_3 - hx_2N'(x_0) - \frac{1}{2}hx_1^2N''(x_0) = 0 & (25) \end{aligned}$$

And from (22) and $c = \gamma - \frac{f(\gamma)}{f'(\gamma)}$, we have

$$\begin{aligned} x_0 &= c = \gamma - \frac{f(\gamma)}{f'(\gamma)} & (26)\\ x_1 &= hN(x_0) & (27)\\ x_2 &= hx_1N'(x_0) & (28)\\ x_3 &= hx_2N'(x_0) + \frac{1}{2}hx_1^2N''(x_0) & (29) \end{aligned}$$

And from (10) (12)& $N(x) = -\frac{g(x)}{f'(\gamma)}$,we have

$$\begin{aligned} N(x_0) &= -\frac{g(x_0)}{f'(\gamma)} = -\frac{f'(x_0)}{f'(\gamma)} & (30)\\ N'(x_0) &= 1 - \frac{f'(x_0)}{f'(\gamma)} & (31)\\ N''(x_0) &= -\frac{f''(x_0)}{f'(\gamma)} & (32) \end{aligned}$$

Note that x is approximated by

$$x = \lim_{m\to\infty} X_m = x_0 + x_1 + ... + x_m \tag{33}$$

For $m = 0$, $x \approx x_0 = \gamma - \frac{f(\gamma)}{f'(\gamma)}$, which implies Newton's method we have discussed.
For $m = 1$, we have

$$\begin{aligned} x &\approx X_1 = x_0 + x_1 = x_0 + hN(x_0)\\ &= \gamma - \frac{f(\gamma)}{f'(\gamma)} - h\frac{g(x_0)}{f'(\gamma)}\\ &= \gamma - \frac{f(\gamma)}{f'(\gamma)} - h\frac{f(x_0)}{f'(\gamma)} \end{aligned}$$

This formulation gives us the following iterative method:

Algorithm 1

For a given x_0, find the approximate solution x_{n+1} by the iterative schemes

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{f'(x_n)}\\ x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} - h\frac{f(y_n)}{f'(x_n)} \end{aligned}$$

For $h=1$, it's exactly two step Newton Method.

For $m=2$, it implies a method with fourth-order convergence.

5.2 Matlab Code for Homotopy Method with $m=2, h=1$

When $n=2, h=1$, then for a given x_0, find the approximate solution x_{n+1} by the iterative schemes

$$
\begin{aligned}
y_n &= x_n - \frac{f(x_n)}{f'(x_n)} \qquad (34)\\
x_{n+1} &= x_n - \frac{f(x_n)+2f(y_n)}{f'(x_n)} + \frac{f(y_n)f'(y_n)}{[f'(x_n)]^2} \qquad (35)
\end{aligned}
$$

Then Matlab code can be writen down as follows:

```
function root=Homotopy(f,a,eps)
if(nargin==2)
 eps=1.0e-6;
end
format long;
fun=diff(sym(f));
fa=subs(sym(f),findsym(sym(f)),a);
dfa=subs(sym(fun),findsym(sym(fun)),a);
if(fa==0)
  root=a;
else
  tol=1;
  r0=a-fa/dfa;
  fr0=subs(sym(f),findsym(sym(f)),r0);
  dfr0=subs(sym(fun),findsym(sym(fun)),r0);
  root=a-(fa+2*fr0)/dfa+(fr0*dfr0)/((dfa)^2);
  while(tol>eps)
     r1=root;
     fx=subs(sym(f),findsym(sym(f)),r1);
     dfx=subs(sym(fun),findsym(sym(fun)),r1);
     r0=r1-fx/dfx;
     fr0=subs(sym(f),findsym(sym(f)),r0);
     dfr0=subs(sym(fun),findsym(sym(fun)),r0);
     root=r1-(fx+2*fr0)/dfx+(fr0*dfr0)/((dfx)^2);
     tol=abs(root-r1);
  end
end
```

This Method is fourth-order convergence, we now calculate the order in a special example with $f(x)=x^3-3x+1$ in the interval $[1,2]$. Then by equation (34) & (35), we will get

$$
\begin{aligned}
g(x) &= x - \frac{\left(-3+\frac{3(1-2x^3)^2}{(3-3x^2)^2}\right)\left(1-\frac{3(1-2x^3)}{3-3x^2}+\frac{(1-2x^3)^3}{(3-3x^2)^3}\right)}{(-3+3x^2)^2}\\
&\quad - \frac{1-3x+x^3+2\left(1-\frac{3(1-2x^3)}{(3-3x^2)}+\frac{(1-2x^3)^3}{(3-3x^2)^3}\right)}{-3+3x^2}
\end{aligned}
$$

We calculate in Matlab that $g'(1.53209)=g''(1.53209)=g'''(1.53209)<10e-5$, where 1.53209 is an approximation of $f(x)=0$, then we can get the method has order 4.

Reference

[1]More J J, Cosnard M. Numerical solution of nonlinear equations[J]. ACM Transactions on Mathematical Software, 1979, 5(1): 64-85.
[2]Sikorski K A. Optimal solution of nonlinear equations[M]. Oxford University Press, USA, 2001.
[3]Richard L. Burden, J. Douglas Faires. Numerical Analysis. Thomson Learning,Inc.2001, 2(3):67-71.
[4]Muhammad Aslam Noor, Iterative Methods for Nonlinear Equations Using Homotopy Perturbation Technique. Applied Mathematics & Information Sciences. 4(2)(2010), 227–235
[5]J. H. He, A new iterative method for solving algebraic equations,Appl. Math. Computation 135(2005), 81–84.