ANALELE UNIVERSITĂȚII
"EFTIMIE MURGU" REȘIȚA
ANUL XV, NR. 1, 2008, ISSN 1453 - 7397

Andrea Minda, Diana Stoica, Mihaela Tomescu

Solving Nonlinear Equations with Mathematica

The aim of this paper is to present numerical methods for solving nonlinear equations and some examples in which we find the roots of this equations using Mathematica.

Keywords: *numerical methods, nonlinear equations, Mathematica*

1. Preliminaries and methods

In this paper we deal with the following problem: finding values of x to satisfy f(x) = 0. Such values are called the roots of the equation.

1.1 The bisection method

We want to find the solution of an equation

$$f(x) = 0 \qquad (1)$$

Theorem 1 If f(x) is continuous on an interval [a , b] and f(a)f(b) < 0 then there exists a point $c \in (a,b)$ such that $f(c) = 0$.

This theorem guarantees that a root exists under those conditions, but it does not tell us the precise value of the root *c*.

The bisection method works by assuming that we know two values *a* and *b* such that $f(a)f(b) < 0$, and works by repeatedly narrowing the gap between *a* and *b* until it closes in on the correct answer.

If $f\left(\frac{a+b}{2}\right) = 0$ then we find a root at $\frac{a+b}{2}$. Otherwise, look at two subsections: $\left(a, \frac{a+b}{2}\right)$ and $\left(\frac{a+b}{2}, b\right)$. By Theorem 1 if $f(a) \cdot f\left(\frac{a+b}{2}\right) < 0$, there must be a

root in the interval $\left(a,\frac{a+b}{2}\right)$, or in the interval $\left(\frac{a+b}{2},b\right)$ when $f\left(\frac{a+b}{2}\right)\cdot f(b)<0$. We continue this procedure until a desired accuracy has been achieved.

1.2. Secant method

The secant method begins by finding two points on the curve of f(x), $M_0(x_0,f(x_0))$ and $M_1(x_1,f(x_1))$, near to a root *c* we seek. The equation to a straight line that passes these two points is

$$\frac{y-f(x_0)}{f(x_1)-f(x_0)}=\frac{x-x_0}{x_1-x_0} \qquad (2)$$

If x_2 is the root of *f(x) = 0*, and the point $M_2(x_2,f(x_2))$ is on this line, then

$$\frac{0-f(x_0)}{f(x_1)-f(x_0)}=\frac{x_2-x_0}{x_1-x_0}$$

and $$x_2=x_1-f(x_1)\frac{x_1-x_0}{f(x_1)-f(x_0)}$$

Because f(x) is not exactly linear, x_2 is not equal to *c*, but it should be closer than either of the two points we begin with.

If we repeat this, we have

$$x_{i+1}=x_i-f(x_i)\frac{x_{i-1}-x_i}{f(x_{i-1})-f(x_i)} \qquad i=1,2,....$$

Under the assumptions that the sequence $(x_n)_{n\geq 1}$ converges to *c*, f(x) is differentiable near *c* and $f'(c)\neq 0$ we obtain:

$$\lim_{n\to\infty}x_{n+1}=\lim_{n\to\infty}x_n-f\left(\lim_{n\to\infty}x_n\right)\frac{\lim\limits_{n\to\infty}x_{n-1}-\lim\limits_{n\to\infty}x_n}{f\left(\lim\limits_{n\to\infty}x_{n-1}\right)-f\left(\lim\limits_{n\to\infty}x_n\right)}$$

or $$c=c-\frac{f(c)}{f'(c)}$$

which gives *f(c) = 0*.

1.3 Newton's method

Newton's method or the Newton-Raphson method is an algorithm for approximating the roots of an equation f(x) = 0). It consists of the following steps:

Step 1. Make a reasonable initial guess as to the location of a solution, which is denoted by x_0.

Step 2. Calculate

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Step 3. If x_0 is sufficiently close to a solution, stop; otherwise, continue this procedure by

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

.....

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

.....

Under the assumptions that the sequence $(x_n)_{n\geq 0}$ converges to c, and that f(x) is differentiable near c with $f'(c) \neq 0$, by taking the limit on both sides of

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

we obtain $$c = c - \frac{f(c)}{f'(c)}$$

which results $f(c) = 0$.

This method requires that the first approximation is sufficiently close to the root c.

The absolute relative aproximate error $|\varepsilon_a|$ is

$$|\varepsilon_a| = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100$$

We compare the absolute relative aproximate error $|\varepsilon_a|$, with the pre-specified relative error tolerance $|\varepsilon_a| > \varepsilon_s$.

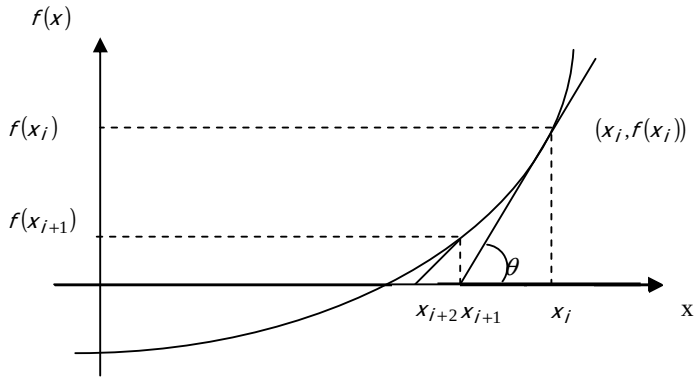

Figure 1.

The secant method is obtained from Newton's method by approximating the derivative of f(x) at two points x_n and x_{n-1} by

$$f'(x) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

Geometrically, Newton's method uses the tangent line and the secant method approximates the tangent line by a secant line.

2. Solving nonlinear equations with Mathematica

Mathematica is the world's most powerful global computation system. First released in 1988, it has had a profound effect on the way computers are used in technical and other fields. In 2007, after many years of development, a major reinvention of *Mathematica* once again transformed how computation is done.

It is often said that the release of *Mathematica* marked the beginning of modern technical computing. Ever since the 1960s individual packages had existed for specific numerical, algebraic, graphical, and other tasks. But the visionary concept of *Mathematica* was to create once and for all a single system that could handle all the various aspects of technical computing--and beyond--in a coherent and unified way.

At first, Mathematica's impact was felt mainly in the physical sciences, engineering, and mathematics. But over the years, Mathematica has become

important in a remarkably wide range of fields, technical and otherwise. Mathematica is used today throughout the sciences--physical, biological, social, and other--and counts many of the world's foremost scientists among its enthusiastic supporters. In engineering, Mathematica has become a standard for both development and production, and by now many of the world's important new products rely at one stage or another on Mathematica in their design

Mathematica is also heavily used in education, and there are now many hundreds of courses--from high school to graduate school--based on it. In addition, with the availability of student versions Mathematica has become a popular and prestigious tool for students around the world.

Built into *Mathematica* is the world's largest collection of both numerical and symbolic equation solving capabilities—with many original algorithms, all automatically accessed through a small number of exceptionally powerful functions. *Mathematica*'s symbolic architecture allows both equations and their solutions to be conveniently given in symbolic form, and immediately integrated into computations and visualizations.

Solve — exact solutions to equations and systems
NSolve — general numerical solutions to equations and systems
FindRoot — numerically find local roots of equations

DSolve — exact solutions to differential equations
NDSolve — numerical solutions to differential equations

RSolve — exact solutions to recurrence equations

FindInstance — find particular solutions to equations and inequalities
Reduce — reduce equations and inequalities

LinearSolve — solve linear systems in matrix form

ContourPlot, **ContourPlot3D** — plot solution curves and surfaces
RegionPlot, **RegionPlot3D** — plot regions satisfied by inequalities

NSolve gives as a general way to find numerical approximations to the solutions of polynomial equations. Finding numerical solutions to more general equations, however, can be much more difficult, FindRoot gives as a way to search for a numerical root of a function or a numerical solution to an arbitrary equation, or set of equations.

In trying to find a solution to an equation, FindRoot starts at the point you specify, and then progressively tries to get closer and closer to a solution. Even if the equations have several solutions, FindRoot always returns the first solution it finds. Which solution this is will depend on what starting point we chose. If we

start sufficiently close to a particular solution, FindRoot will usually return that solution.

Example 1

The ball in figure 2 has a gravity of 0.6 and has a radius of 5.5 cm. Find the distance to which the ball will get submerged floating in water (See Figure 2).

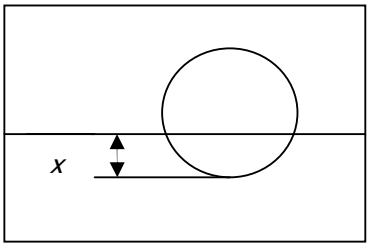

Figure 2. Floating ball

The equation that gives the depth 'x' to which the ball is submerged under water is given by

$$x^3 - 0.165x^2 + 3.993 \cdot 10^{-4} = 0$$

Using the Newton method of finding roots of equations to find the depth 'x' to which the ball is submerged under water. We conduct three iterations to estimate the root of above equation and find the absolute relative approximate error at the end of each iteration.

$f(x) = x^3 - 0.165x^2 + 3.993 \cdot 10^{-4} \qquad f'(x) = 3x^2 - 0.33x$

We assume that the initial guess of the root of $f(x) = 0$ is $x_0 = 0.05$

Iteration 1 :The estimate of the root is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0.05 - \frac{1.118 \cdot 10^{-4}}{-9 \cdot 10^{-3}} = 0.06242$$

The absolute relative error $|\varepsilon_a|$ at the end of iteration 1 is $|\varepsilon_a| = 19.89\%$

Iteration 2 : The estimate of the root is

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.06242 - \frac{-3.97781 \cdot 10^{-7}}{-8.90973 \cdot 10^{-3}} = 0.06238$$

The absolute relative error $|\varepsilon_a|$ at the end of iteration 2 is $|\varepsilon_a| = 0.07157\%$

Iteration 3 : The estimate of the root is

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 0.06238 - \frac{4.42937 \cdot 10^{-11}}{-8.91171 \cdot 10^{-3}} = 0.06238$$

The absolute relative error $|\varepsilon_a|$ at the end of iteration 3 is $|\varepsilon_a| = 0\%$

The Mathematica sequence for finding the root of $x^3 - 0.165x^2 + 3.993 \cdot 10^{-4} = 0$ is in figure 3.

Example2.

We want to find the root of the equation $\left(\frac{y}{2}\right)^2 - \sin y - 0.5 = 0$. Looking at the graph of $f(y) = \left(\frac{y}{2}\right)^2 - \sin y - 0.5$ we see that this equation have 2 real roots. The equation has several solutions. If you start at a different *x*, FindRoot may return a different solution.(see figure 3). The Mathematica sequence for finding the roots of this equatoin is:

```
FindRoot[(y/2)^2-Sin[y]-0.5==0,{y,1}]
```

{y -> -0.462799}

```
FindRoot[(y/2)^2-Sin[y]-0.5==0,{y,2}]
```

{y -> 2.25708}

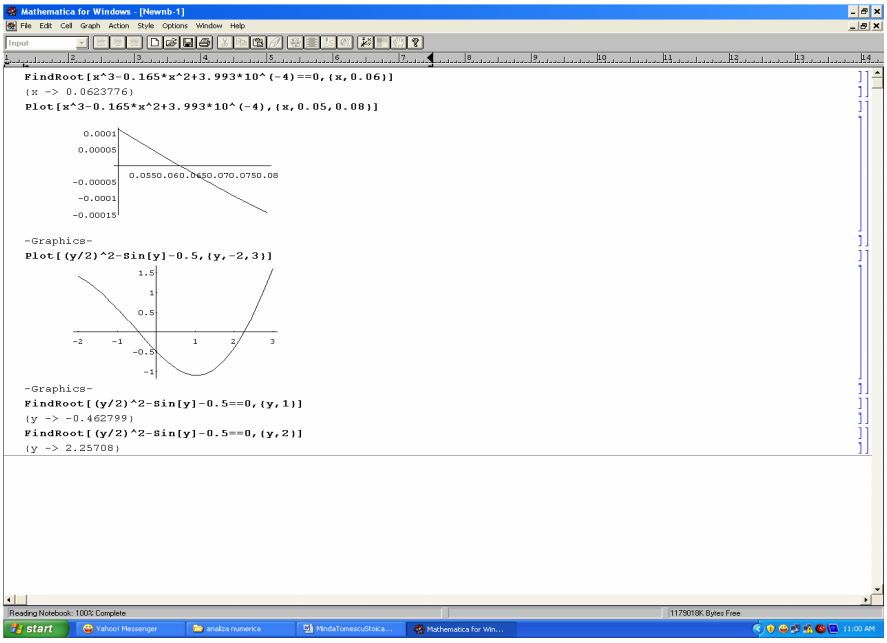

Figure 3

References

[1] Kaw, A., An interactive e-book for illustrating Newton-Raphson method of solving nonlinear equations, Holistic Numerical Methods ,Institute College of Engineering,University of South Florida.

[2] Maruster,S., Metode numerice in rezolvarea ecuatiilor neliniare, Ed. Tehnica , Bucuresti,1981

[3]http://reference.wolfram.com/mathematica/guide/EquationSolving.html

Addresses:

- Lect. univ. drd. Andrea Minda, "Eftimie Murgu" University of Reşiţa, Piaţa Traian Vuia, nr. 1-4, 320085, Reşiţa, andreaminda@yahoo.com
- Asist. univ. drd. Mihaela Tomescu, University of Petrosani , mihaela_tomescu2000@yahoo.com
- Asist. univ. drd. Diana Stoica, University "Politehnica" of Timişoara, The Faculty of Engeneering of Hunedoara, Str. Revoluţiei, Nr.5, 331128, stoicadianna@yahoo.com