Nonlinear Equations

Fotios Kasolis
PH.D. IN COMPUTATIONAL SCIENCE & ENGINEERING
kasolis@uni-wuppertal.de

Introduction

Many engineering problems are inherently nonlinear. In contrast to linear systems of equations, which can be analytically solved, nonlinear equations can be rarely solved by hand, while it is even difficult to verify whether a solution exists, since there is no general formula. Hence, numerical methods are utilized even in the scalar case. As an example, we consider the function $f : \mathbb{R} \to \mathbb{R}$ with values $f(x) = x + 2\cos x$ and we seek one of its zeros using the following Matlab code snippet.

```
>> format long;
>> f  = @(x) x + 2.0*cos(x); % anonymous function definition
>> x0 = 0.0;                 % initial guess
>> xz = fzero(f, x0)
xz =
  -1.029866529322259
```

Here, we consider the equations' set

$$f_1(x_1, \ldots, x_n) = 0, \quad f_2(x_1, \ldots, x_n) = 0, \quad \ldots, \quad f_n(x_1, \ldots, x_n) = 0,$$

where f_i are n nonlinear functions in n unknowns x_i. Using vector notation, this system is commonly written in the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

where

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top \quad \text{and} \quad \mathbf{f} = (f_1, f_2, \ldots, f_n)^\top.$$

Example. Consider the system of two simultaneous nonlinear equations in two unknowns, x_1 and x_2, defined by

$$x_1^2 + x_2^2 - 1 = 0, \quad 5x_1^2 + 21x_2^2 - 9 = 0.$$

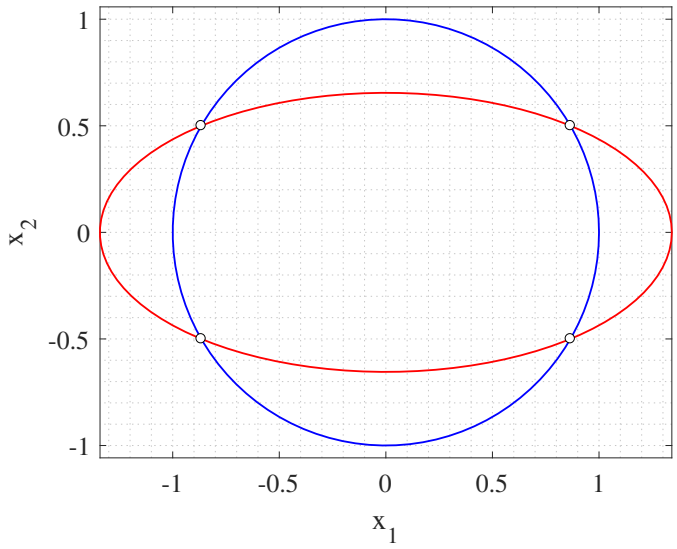
Here, $\mathbf{x} = (x_1, x_2)^\top$ and $\mathbf{f} = (f_1, f_2)^\top$ with

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1, \quad f_2(x_1, x_2) = 5x_1^2 + 21x_2^2 - 9.$$

The equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ has four solutions; that, is,

$$\boldsymbol{\alpha}_1 = (-\sqrt{3}/2, 1/2)^\top, \quad \boldsymbol{\alpha}_2 = (\sqrt{3}/2, 1/2)^\top,$$
$$\boldsymbol{\alpha}_3 = (-\sqrt{3}/2, -1/2)^\top, \quad \boldsymbol{\alpha}_4 = (\sqrt{3}/2, -1/2)^\top.$$

The curves defined by $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$ are depicted below. The four solutions correspond to the four points of intersection of the two curves in the figure.



Fixed point iterations

The roots of the prototype system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ are determined with so-called **iterative methods**; that is, methods that improve an initial approximation $\mathbf{x}_0$ and introduce additional error, besides rounding. Since any system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ can be written as

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{x} - \mathbf{g}(\mathbf{x}) = \mathbf{0},$$

a scheme naturally suggested is given by

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k), \quad k = 0, 1, 2, \ldots.$$

Hence, an iterative method can be written as a recursion formula for some function $\mathbf{g}$ and an initial approximation $\mathbf{x}_0$. It can be shown that, under certain conditions[1], $\mathbf{g}$ has a unique **fixed point** $\mathbf{x}_*$; that is, a point satisfying the equation $\mathbf{g}(\mathbf{x}) = \mathbf{x}$.

Example. We consider the system stated in the previous example. Let us suppose that we need to find the solution of the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in the first quadrant of the (x_1, x_2)-coordinate system. Let us write the two equations as

$$x_1 = \sqrt{1 - x_2^2}, \quad x_2 = \frac{1}{\sqrt{21}}\sqrt{9 - 5x_1^2},$$

and define $g_1(x_1, x_2)$ and $g_2(x_1, x_2)$ as the right-hand sides of these, respectively. Then, we consider the recursion

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k), \quad k = 0, 1, 2, \ldots,$$

with suitably chosen initial guess $\mathbf{x}_0$ and $\mathbf{g} = (g_1, g_2)^\top$.

Newton's method

Let us consider the equation $f(x) = 0$, where $f : \mathbb{R} \to \mathbb{R}$ is a nonlinear, differentiable function. To construct a method that exploits function and derivative values, we Taylor-expand f around $x + \Delta x$ and we neglect second order terms; that is, we obtain the equation of the tangent to the graph of f at x,

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x.$$

If $f(x + \Delta x) \approx 0$ and $f'(x) \neq 0$, then $\Delta x \approx -f(x)/f'(x)$. Given Δx, we construct an improving sequence (x_k) using the recursion formula

$$x_{k+1} = x_k + \Delta x = x_k - [f'(x)]^{-1} f(x), \quad k = 0, 1, 2, \ldots.$$

This method is known as **Newton's method** and corresponds to computing a zero of f by locally replacing f with its tangent line. In general, Newton's method does

[1] If $\mathbf{g}$ carries a closed and bounded set into itself and if the mapping is contracting, that is, if $\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for some norm, Lipschitz constant $L < 1$, and all $\mathbf{x}$, $\mathbf{y}$.

not converge for all possible choices of x_0, but only for those x_0 in the vicinity of the exact solution α. In practice, a possible initial value x_0 can be obtained by resorting to a few iterations of another method, such as the bisection method. If x_0 is properly chosen and $f'(\alpha) \neq 0$, then the Newton method converges. Newton's method is said to converge quadratically, since for sufficiently large values of k the error at step $k+1$ behaves like the square of the error at step k multiplied by a constant which is independent of k, more precisely

$$\lim_{k\to\infty} \frac{x_{k+1}-\alpha}{(x_k-\alpha)^2} = C \quad \left(= \frac{f''(\alpha)}{2f'(\alpha)}\right).$$

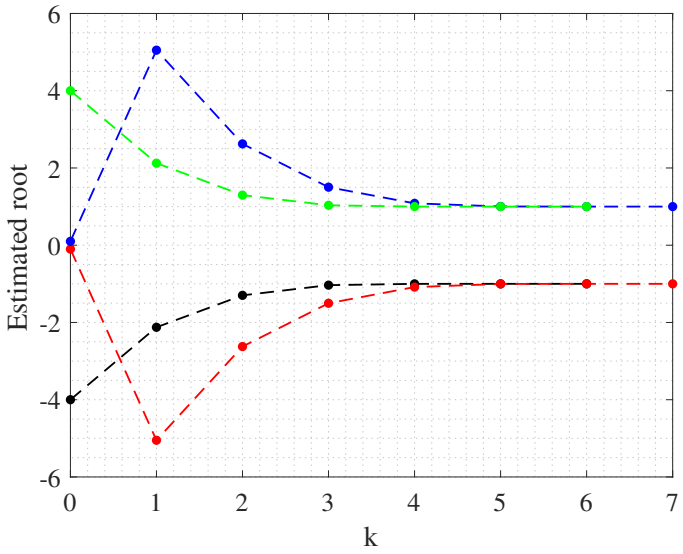
To terminate Newton's iterations, we test the condition $|f(x_k)| < \epsilon$, where $0 < \epsilon \ll 1$, at each iteration, and we stop when it evaluates to true.

Example. Consider the function $f : \mathbb{R} \to \mathbb{R}$ with values $f(x) = x^2 - 1$. The derivative of f is $f'(x) = 2x$, and the exact zeros of f are ± 1. A naive Matlab implementation `newton` computes a root of $f(x) = 0$, where $\epsilon = 10^{-8}$ and the maximum number of allowed iterations is 100.

```
>> f  = @(x) x^2 - 1;                 % function
>> fp = @(x) 2*x;                     % derivative
>> x0 = -4.0; x1 = newton(f, fp, x0); % x0 is initial guess
>> x0 = -0.1; x2 = newton(f, fp, x0);
>> x0 = +0.1; x3 = newton(f, fp, x0);
>> x0 = +4.0; x4 = newton(f, fp, x0);
```

where a naive `newton` function is implemented in the following Matlab function file `newton.m`.

```
function [ x ] = newton(f, fp, x0)
tol = 1.0;
noi = 100;
x = zeros(noi, 1);
x(1) = x0;
k = 1;
while ( abs(tol)>1e-8 && k<noi )
  x(k+1) = x(k) - f(x(k))/fp(x(k));
  tol = abs(f(x(k+1)));
  k = k + 1;
end
x = x(1:k);
```



Now, we consider the multidimensional scenario. For that purpose, we let

$$\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + D\mathbf{f}(\mathbf{x})\Delta\mathbf{x}$$

be a linear approximation of the vector function $\mathbf{f}$ around $\mathbf{x}$, where

$$\mathbf{J}(\mathbf{x}) \equiv D\mathbf{f}(\mathbf{x}) = \left(\frac{\partial f_i}{\partial x_j}\right)(\mathbf{x})$$

is the so-called **Jacobian matrix**. We wish to find the direction $\Delta\mathbf{x}$ towards which $\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) \approx \mathbf{0}$, thus,

$$\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x} = \mathbf{0} \Leftrightarrow \mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}).$$

If $\mathbf{J}(\mathbf{x})$ is non-singular, the solution vector $\Delta\mathbf{x}$ is uniquely defined, that is, $\Delta\mathbf{x} = -\mathbf{J}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})$, and in this case, an improved estimate $\mathbf{x}_{k+1}$ is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{J}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x}),$$

without the need to explicitly form the inverse of the Jacobian[2]. Given tolerance $0 < \epsilon \ll 1$, Newton's iterations are terminated when the condition $\|\mathbf{f}(\mathbf{x}_k)\| \leq \epsilon$ evaluates to true, for some k.

[2]The step $\Delta\mathbf{x}$ is obtained by solving the linear system $\mathbf{J}\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x})$.

Example. We close this theme with an example which illustrates the application of Newton's method. Consider the simultaneous equations

$$\begin{aligned} f_1(x,y,z) &\equiv x^2+y^2+z^2-1=0, \\ f_2(x,y,z) &\equiv 2x^2+y^2-4z-1=0, \\ f_3(x,y,z) &\equiv 3x^2-4y+z^2=0. \end{aligned}$$

Letting $\mathbf{f}=(f_1,f_2,f_3)^\top$ and $\mathbf{x}=(x,y,z)^\top$, the aim of the exercise is to determine the solution to the equation $\mathbf{f}(\mathbf{x})=\mathbf{0}$ contained in the first octant

$$\{(x,y,z)\in\mathbb{R}^3 : x>0, y>0, z>0\}.$$

Note that the Jacobian matrix of $\mathbf{f}$ at $\mathbf{x}\in\mathbb{R}^3$ is

$$\mathbf{J}(\mathbf{x})=\begin{pmatrix} 2x & 2y & 2z \\ 4x & 2y & -4 \\ 6x & -4 & 2z \end{pmatrix}.$$

Since the first equation represents a sphere of radius one centered at the origin, and the second and third equations describe elliptic paraboloids whose axes are aligned with the coordinate semiaxis $(0,0,z)$, $z\geq 0$, and $(0,y,0)$, $y\geq 0$, respectively, the point of intersection of the three surfaces belong to $[0,1]^3$. Choosing $\mathbf{x}_0=0.5(1,1,1)^\top$, we get $\mathbf{f}(\mathbf{x}_0)=(-0.25,-1.25,-1.00)^\top$ and

$$\mathbf{J}(\mathbf{x}_0)=\begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{pmatrix}.$$

On solving the system of linear equations

$$\mathbf{J}(\mathbf{x}_0)(\mathbf{x}_1-\mathbf{x}_0)=-\mathbf{f}(\mathbf{x}_0)$$

for $\mathbf{x}_1-\mathbf{x}_0$, we find $\mathbf{x}_1=(0.875,0.500,0.375)^\top$. Similarly,

$$\mathbf{x}_2=(0.78981,0.49662,0.36993)^\top,$$

$$\mathbf{x}_3=(0.78521,0.49662,0.36992)^\top.$$

As $\mathbf{f}(\mathbf{x}_3)=10^{-5}\cdot(1,4,5)^\top$, the vector $\mathbf{x}_3$ can be thought of as a satisfactory approximation to the required solution; after rounding to four decimal digits, we have that

$$x=0.7852, \quad y=0.4966, \quad z=0.3699.$$

Problem set

1. Let a sequence (x_n) of real numbers be defined by $x_{k+1} = x_k(2 - dx_k)$, where $d \neq 0$ is a known real number. Assume that the sequence converges to a nonzero number. (a) To what number does the sequence converge? (b) Show that the convergence rate is quadratic.

2. The Mandelbrot set is the set of complex numbers c for which the sequence generated by iterating the complex quadratic map

$$z_{k+1} = z_k^2 + c, \quad z_0 = 0, \quad k = 0, 1, 2, \ldots,$$

remains bounded; in particular, $|z_n| \leq 2$ as $n \to \infty$. (a) Write the quadratic mapping $z_{k+1} = f(z_k)$ as a system. (b) Construct and plot the Mandelbrot set on $[-2, 1] \times [-1, 1]$ of the complex plane.

3. For the 3×3 system in the example of this theme, write a Matlab function that returns the function values and the associated Jacobian matrix at a particular point. Implement both the exact Jacobian and a finite difference approximation.

4. Implement a multidimensional Newton method in Matlab, and use your implementation, as well as `fsolve`, to solve the 3×3 system presented in the example of this theme. Use three different initial guesses and comment on the results.