Methods for Solving Nonlinear Systems of Equations

Wanjie Zhang

1135940

University of Washington

August 12, 2013

Abstract

Nonlinear systems of equations are very important in many aspects, for instance, in chemistry, physics and economy. In this project, several methods of solving nonlinear systems would be introduced. The nonlinear systems problems are usually hard to solve. Methods are going to be used to simplify the nonlinear systems, including local linearization. Also, the iterative methods would be used to solve nonlinear problems,including Newton's method, SOR and etc.. Classic problems would be raised and solved after the introduction of each method. In comparing efficiency of iterative methods, the rate of convergence (ROC) would be introduced.

1 Introduction

Solving of nonlinear systems of equation is a very important part in mathematic and has a wide range of application in various aspects, including chemistry, physics and economy. Some of nonlinear problems could be transformed into a linear, however some could not. Therefore, finding methods to solve nonlinear systems is crucial. The main aspects about nonlinear systems would be overviewed. Some definition of terms would be mentioned. The local linearization of nonlinear systems would be introduced in this project. For some problems that cannot be linearized, iterative methods would be used and discussed.

2 Background

2.1 Problem Overview

Let $F : R^n \to R^n$ be a nonlinear mapping from the n-dimensional real linear space R^n into itself. Our interest is in the method for the compitation of solutions of the systems of n equations in n variables.

$$Fx = 0$$

We need to consider following problem in solving the equation above:
(a) Do solutions exist in a specified subset of the domain of F?
(b) How many solutions are there in such a set?
(c) How do the solutions vary under small changes to the problems?
After discussing these simplified methods, the iterative methods would be discussed.

2.2 Definition of terms used in Iterative Methods

2.2.1 Q-factor and R-factor

Let's define the Q-factor and R-factor first First we need to define the terms I will use in iterative methods; Q Cfactor and R-factor.

DEFINITION for a sequence $\{x^k\} \subset R^n$ with limit $x^* \in R^n$ and any $p \in [1, \infty)$

$$R_p\{x^k\} = \begin{cases} \limsup_{k\to\infty} \|x^k - x^*\|^{\frac{1}{k}}, & \text{if p =1;} \\ \limsup_{k\to\infty} \|x^k - x^*\|^{\frac{1}{p^k}}, & \text{if p ¿1.} \end{cases}$$

are the root-convergence factors (R-factors) of $\{x^k\}$
Moreover, $R_p(J, x^*) = sup\{R_p\{x^k\} : \{x^k\} \in C(J, x^*)\}$

Similarly, we have:
DEFINITION for a sequence $\{x^k\} \subset R^n$with limit $x^* \in R^n$ and any $p \in [1, \infty)$

$$Q_p\{x^k\} = \begin{cases} \limsup\limits_{k\to\infty} \frac{\|x^{k+1}-x^k\|}{\|x^k-x^*\|^p}, & \text{if } x^k \neq x^* \forall k \geq k_0; \\ 0, & \text{if } x^k \neq x^* \forall k = k_0; \\ \infty, & \text{otherwise} \end{cases}$$

are the quotient-convergence factors (Q-factors) of $\{x^k\}$ with respect to the particular norm. Moreover,

$$Q_p(J, x^*) = sup\{R_p\{x^k\} : \{x^k\} \in C(J, x^*)\}$$

are the Q-factors of an iterative process J with limit x^*

The Q- and R- factors can be used to compare the rate of convergence of different iterative processes.A comparison of two iterative processes, J_1 and J_2 , in terms of the R-measure proceeds as follows.
First compare the R-orders $o_i = O_R(J_i, x^*)$, i = 1, 2; the process with the larger R-order is R-faster than the other one. For $o_l = o_2 = p$ compare the R-factors $\gamma_i = R_p(J_i, x^*)$, i = 1,2; if one of these numbers is smaller, the corresponding process is R-faster. We do the similar thing for Q-measure.

2.2.2 Rate of Convergence

General DEFINITION for rate of convergence:
A sequence $x^k, k = 0, 1, 1...inR^n$ converges linearly to $x^* \in R^n$ if

$$\exists L < 1 : \|x^{k+1} - x^*\| \leq L\|x^k - x^*\|, \forall k \in N_0$$

3 Linearization

3.1 Local linearization of Nonlinear Systems

For a nonlinear state-space systems of the form

$$x = f(x, u)$$

$$y = h(x, u)$$

the locally linearized models is given by

$$\tilde{x} = A\tilde{x} + B\tilde{u}$$

$$\tilde{y} = C\tilde{x} + D\tilde{u}$$

Then we have:

$$A = \frac{\partial f}{\partial x} \mid_n$$
$$B = \frac{\partial f}{\partial u} \mid_n$$
$$C = \frac{\partial h}{\partial x} \mid_n$$
$$D = \frac{\partial h}{\partial u} \mid_n$$

where the Jacobian $\frac{\partial g}{\partial t}$ for a vector of functions

$g(t) = \begin{pmatrix} g_1(t_1, t_2, ..., t_n) \\ g_2(t_1, t_2, ..., t_n) \\ \vdots \\ g_n(t_1, t_2, ..., t_n) \end{pmatrix}$

in variables $t_1, t_2, ..., t_n$ is

$\frac{\partial g}{\partial t} = \begin{pmatrix} \frac{\partial g_1}{\partial t_2} & \frac{\partial g_1}{\partial t_1} & \cdots & \frac{\partial g_1}{\partial t_n} \\ \frac{\partial g_2}{\partial t_2} & \frac{\partial g_2}{\partial t_1} & \cdots & \frac{\partial g_2}{\partial t_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial g_m}{\partial t_2} & \frac{\partial g_m}{\partial t_1} & \cdots & \frac{\partial g_m}{\partial t_n} \end{pmatrix}$

The function Linearize attempts to find such an approximation.
Linearize $[f, \{x, x_n\}, \{u, u_n\}]$
find a linearized state-space approximation to the function f in the state variable x and input variable u in the vicinity of the operating point x_n, u_n

Linearize $[f, \{\{x_1, x_{1n}\}, \{x_2, x_{2n}\}, ...\}, \{\{u_1, u_{1n}, \{u_2, u_{2n}\}, ...\}]$
find a linearized state-space approximation to the function f in the state variable x and input variable u in the vicinity of the operating point x_{in}, u_{in}

Linearize $[f, h, \{\{x_1, x_{1n}\}, \{x_2, x_{2n}\}, ...\}, \{\{u_1, u_{1n}, \{u_2, u_{2n}\}, ...\}]$
find a linearized state-space approximation to the vectors f and h

3.2 Linearization Concepts

Let $x^k \in E$ be a current iterate when solving $Fx = 0$. the idea of a linearization method is to construct an affine approximation

$$L_k : R^n \to R^n, L_k x = A_k(x - x^k) + Fx^k, A_k \in L(R^n)$$

that agrees with F at x^k, and to use a solution of $L_k x = 0$ as the next iterate. We shall not consider here the situation when some or all of the matrices are allowed to be singular, and assume always that all A_k , $k \geq 0$ are invertible. Then the resulting (nonsingular) linearization method becomes

$$x^{k+1} = x^k - A_k^{-1}Fx^k, k = 0, 1....$$

The simplest linearization methods are the chord methods where all matrices A_k are identical. In other words, chord methods have the form:

$$x^{k+1} = x^k - A_k^{-1}Fx^k, k = 0, 1....$$

A special case of the Picard iteration arises if F has the from $Fx = Bx - Gx$ with nonsingular $B \in L(R^n)$ and a nonlinear mapping G:$E \subset R^n \to R^n$. Here the chord method with the choice A=B becomes:

$$x^{k+1} = A^{-1}Gx^k, k = 0, 1....$$

which, for B=I; that is , for the fixed-point equation x=Gx, is the classical method of successive approximations

$$x^{k+1} = Gx^k, k = 0, 1....$$

for the convergence of the simple methods of the form we also have the contraction-mapping theorem, which in our setting may be phrased as follows
$THEOREM$ Suppose that G:$E \subset R^n \to R^n$ maps a closed set $C \subset E$ into itself and that there exists a constant $\alpha \in (0, 1)$ such that

$$\|Gy - Gx\| \leq \alpha\|y - 1\|, \forall x, y \in C$$

Then G has a unique fixed point $x^* = Gx^*$ in C and for any $x^0 \in C$ the iterates converge to x^* and satisfy

$$\|x^k - x^*\| \leq \frac{\alpha^k}{a - \alpha}\|x^1 - x^0\|, k \geq 0$$

Proof. Any two fixed points $x^*, y^* \in C$ satisfy

$$\|y^* - x^*\| = \|Gy^* - Gx^*\| \leq \alpha\|y^* - x^*\|$$

which, because $\alpha < 1$, implies that $y^* = x^*$. Since GC⊂ C, the sequence $\{x^k\}$ of $x^{k+1} = Gx^k$ is well defined and remains in C. For any k,m> 0we obtain

$$\|x^{k+m+1} - x^k\| \leq \sum_{j=0}^{m} \|x^{h+j+1} - x^{k+j}\|$$

$$\leq \sum_{j=0}^{m} \alpha^j\|x^{k+1} - x^k\| \leq \frac{1}{1 - \alpha}\|x^{k+1} - x^k\|$$

$$\leq \frac{\alpha^k}{1 - \alpha}\|x^1 - x^0\|$$

which shows that $\{x^k\}$ is a cauchy sequence. Thus $lim_{k\to\infty}x^k = x^*$ exists and the closedness of C implies that $x^* \in C$ From $\|x^k - x^*\| \leq \frac{\alpha^k}{a-\alpha}\|x^1 - x^0\|, k \geq 0$, it follows that G is continuous whence $x^{k+1} = A^{-1}Gx^k, k = 0, 1....$ gives $x^* = Gx^*$. The error estimate $\|x^k - x^*\| \leq \frac{\alpha^k}{a-\alpha}\|x^1 - x^0\|, k \geq 0$, is a sequence od the above proof for $m \to \infty$

4 Iterative Methods

4.1 Newton Methods

4.1.1 Form of Newton Method

At first, I will give a general idea about Newton Method
Lets suppose that we want to approximate the solution to f(x)=0 and lets also suppose that we have somehow found an initial approximation to this solution say, x_0. This initial approximation is probably not all that good and so wed like to find a better approximation. This is easy enough to do. First we will get the tangent line to f(x) at x_0.
Now, take a look at the graph below.

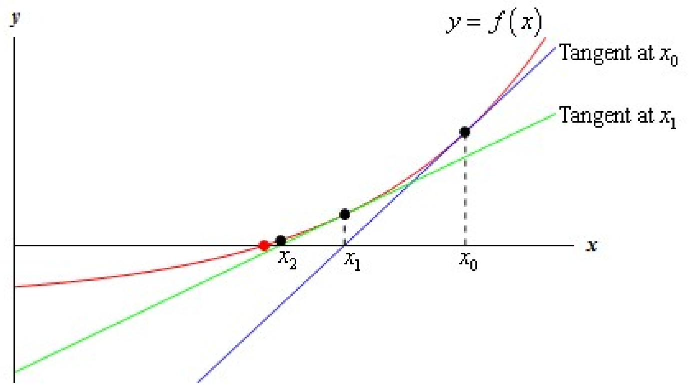


Figure 1: General Newton Method

The blue line (if youre reading this in color anyway) is the tangent line at x0. We can see that this line will cross the x-axis much closer to the actual solution to the equation than x_0 does. Lets call this point where the tangent at x_0 crosses the x-axis x_1 and well use this point as our new approximation to the solution.
So, how do we find this point? Well we know its coordinates,(x_1,0) , and we know that its on the tangent line so plug this point into the tangent line and solve for x1 as follows,

$$0 = f(x_0) + f'(x_0)(x_1 - x_0)$$

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

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

So, we can find the new approximation provided the derivative isnt zero at the original approximation. Now we repeat the whole process to find an even better approximation. We form up the tangent line to f(x) at x_1 and use its root, which well call x2, as a new approximation to the actual solution. If we do this we will arrive at the following formula.

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

This point is also shown on the graph above and we can see from this graph that if we continue following this process will get a sequence of numbers that are getting very close the actual solution. This process is called Newtons Method.
Here is the general Newtons Method

We have:

$$N : x^{k+1} = x^k - D(x^k)^{-1} F x^k, k = 0, 1, ...$$

The execution of Newton methods requires the following:
(a) evaluation of the n components of Fx^k
(b) evaluation of the n^2 elements of A_k
(c) numerical solution of the $n \times n$ linear system
(d) work needed to handle the memory set M_{k+1}

4.1.2 Convergence of Newton Methods

Next,we are going to check whether Newton Method is convergent or not and how fast would this method convergence. With the formulation we draw above, we can expand function F about x^*

$$F(x) = F^* + J^*(x - x^*) + O((x - x^*)^2)$$
$$F(x) = J^*(e) + O[e^2]$$

gives:

$$O[e^2] = J^*(e^{i+1})$$

Then we can see the convergence is quadratic.

4.1.3 Example in Chemistry

ex: chemical reaction:

$$2A + B \Leftrightarrow C$$
$$A + D \Leftrightarrow C$$

a_0, b_0, d_0: initial concentrations (moles/liter) in chemical reactor(known)
c_1, c_2: equilibrium concentrations of C produced by each reaction(unknown)
k_1, k_2:equilibrium reaction constants(known)
These variables are related by the las of mass action.

$$\frac{c_1 + c_2}{(a_0 - 2c_1 - c_2)^2 (b_0 - c_1)} = k_1$$
$$\frac{c_1 + c_2}{(a_0 - 2c_1 - c_2)(d_0 - c_2)} = k_2$$

We used the knowledge we have learned about Newton's Method:
Given(x_n, y_n), we want to find (x_{n+1}, y_{n+1})

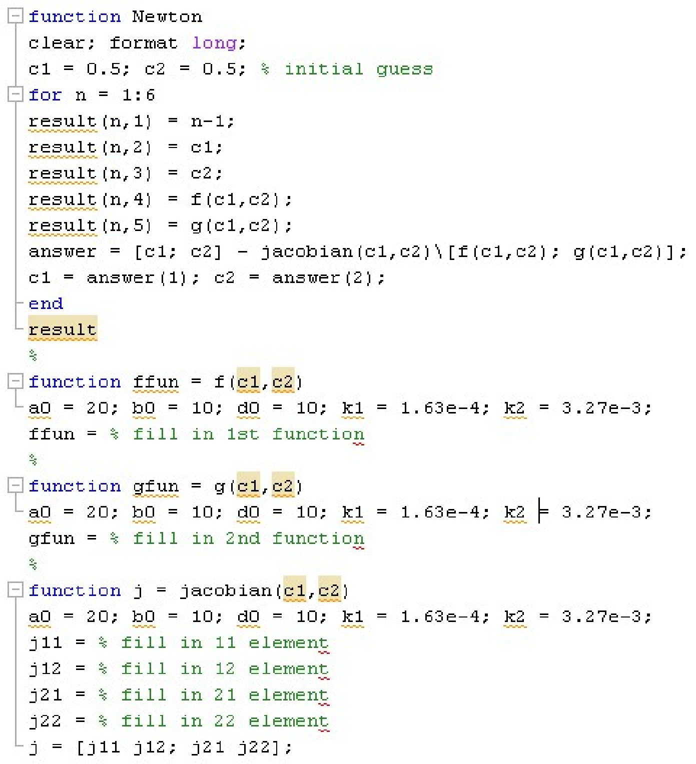
$$0 = f(x_{n+1}, y_{n+1}) = f(x_n, y_n) + \frac{\partial f}{\partial x}(x_n, y_n)(x_{n+1} - x_n) + \frac{\partial f}{\partial y}(x_n, y_n)(y_{n+1} - y_n)$$
$$0 = g(x_{n+1}, y_{n+1}) = f(x_n, y_n) + \frac{\partial g}{\partial x}(x_n, y_n)(x_{n+1} - x_n) + \frac{\partial g}{\partial y}(x_n, y_n)(y_{n+1} - y_n)$$

Then we have:

$$\begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix} |_{x_n, y_n} \cdot \begin{pmatrix} x_{n+1} - x_n \\ y_{n+1} - y_n \end{pmatrix} = \begin{pmatrix} -f(x_n, y_n) \\ -g(x_n, y_n) \end{pmatrix}$$

With the equation mentioned above, we can easily solve the chemical problem with the Newton's method.

4.1.4 Form of Matlab code using Newton methods



```
function Newton
clear; format long;
c1 = 0.5; c2 = 0.5; % initial guess
for n = 1:6
result(n,1) = n-1;
result(n,2) = c1;
result(n,3) = c2;
result(n,4) = f(c1,c2);
result(n,5) = g(c1,c2);
answer = [c1; c2] - jacobian(c1,c2)\[f(c1,c2); g(c1,c2)];
c1 = answer(1); c2 = answer(2);
end
result
%
function ffun = f(c1,c2)
a0 = 20; b0 = 10; d0 = 10; k1 = 1.63e-4; k2 = 3.27e-3;
ffun = % fill in 1st function
%
function gfun = g(c1,c2)
a0 = 20; b0 = 10; d0 = 10; k1 = 1.63e-4; k2 = 3.27e-3;
gfun = % fill in 2nd function
%
function j = jacobian(c1,c2)
a0 = 20; b0 = 10; d0 = 10; k1 = 1.63e-4; k2 = 3.27e-3;
j11 = % fill in 11 element
j12 = % fill in 12 element
j21 = % fill in 21 element
j22 = % fill in 22 element
j = [j11 j12; j21 j22];
```

Figure 2: Newton Method

4.2 SOR

We are going to introduce a method that would generally e most efficient in iterative method. According to our experience in solving the linear system, the SOR method choose the most efficient ω_k to produce the iterative step. With this certain ω_k, the SOR would converge much faster than Jocobian and GS method. Next, we are going to apply SOR to nonlinear system.

We are going to focus on SOR–Steffensen-Newton method to have a basic expression of SOR method.
To solve $F(x) = 0$ for $F = (F_1, F_2, ...F_n)^t$ is defined by solving the following equation for x_t

$$F_t(x_1^{k+1}, ...x_{t-1}^{k+1}, x_t, x_{t+1}^k, ...x_n^k) = 0$$

Then we have:

$$x_l = y_l^n - \frac{F_t(x_1^{k+1}, ...x_{t-1}^{k+1}, y_l^n, x_{t+1}^k, ...x_n^k)}{F_{tt}(x_1^{k+1}, ...x_{t-1}^{k+1}, y_l^n, x_{t+1}^k, ...x_n^k)}$$

where

$$y_l^n = x_l^k - \frac{F_l(x^{k,l-1})}{\frac{F_l(x^{k,l-1}+F_l(x^{k,l-1})e^l)-F_l(x^{k,l-1})}{F_l(x^{k,l-1})}}$$

Finally we have:

$$x_l^{k+1} = y_l^n + \omega(x_l - y_l^n)$$

And with the limited time, I would directly use the conclusion from *From Linear to Nonlinear Iterative Methods*(M. N. Vrahatis, G.D. Magoulas, 2002) that the SOR-SN method is convergent.
Also, with reference to the *SOR- Steffensen-Newton Method to Solve Systems of Nonlinear Equations*(M. T. Darvishi , Norollah Darvishi,2012), the SOR-SN method is were better in number of iterations when comparing to SOR-Newton, SOR-Steffensen and SOR-Secant methods.

4.3 Secant Type

As an extension of one-dimensional Secant Methods, we could represent the form of Secant Method in higher dimensions as:

$$L_k x = A_k x + a^k, A_k \in L(R^n), a^k \in R^n$$

THEOREM *For given $x^{k,j}, y^{k,j} \in R^n, j = 0, 1, ..., n$, there exists a unique function such that $L_k x^{k,j} = y^{k,j}, j = 0, 1, ..., n$, if and only if the $\{x^{k,j}\}$ are in general position. Moreover, A_k is nonsingular if and only if the $\{y^{k,j}\}$ are in general position.*

With the similar step in one dimension, we are going to represent $x^{k,j}$ in the following form:

$$x^{k,j} = x^k + H_k e^j$$

where

$$H_k = (x^{k,1} - x^k, ..., x^{k,n} - x^k)$$

With similar substitution in Newton Methods, we have:

$$x^{k+1} = x^k - J(x^k, H_k)^{-1} F x_k, k = 0, 1....$$

where $J(x, H) = \Gamma H^{-1}, \Gamma = (F(x + He^1) - Fx, ..., F(x + He^n) - Fx)$
We called the new form of x^{k+1} a general secant method. Usually, the secant method should be conducting in the following way:

```
input: $\{k, x^k, M_k\}$
construct $x^{k,1}, ..., x^{k,n}$
evaluate $Fx^k, Fx^{k,1}, ..., Fx^{k,n}$
solve the following:
```

$$0 = \sum_{j=0}^{n} z_j F x^{k,j} = a^k + A_k \sum_{j=0}^{n} z_j x^{k,j}, \sum_{j=0}^{n} z_j = 1$$

```
if solver failed then return: fail
$x^{k+1} = z_0 x^k + \sum_{j=1}^{n} z_j x^{k,j}$
return:$\{x^{k+1}, M_{k+1}\}$
```

5 Future work

After learning all these method above, I have some basic understanding in how to solve nonlinear systems. In future, if there are any chance for me to explore more on this topic, I would choose to study Higher-Order Method.

5.1 Higher-Order Methods

In the linearization method the nonlinear function F is approximated near the kth iterat x^k by an mapping $L_k x = Fx^k + A_k(x - x^k)$ and then a solution of $L_k x = 0$ is used as x^{k+1}. In future I would go to explore the higher-order method to make the iterative methods converges much quicker.

6 Conclusion

With few weeks researching the topic of methods for solving the nonlinear systems of equations, I have learned some basic ways to solve nonlinear systems, which would be quite useful in my further study of mathematics. In this project, with the knowledge I learned about the nonlinear systems, I used matlab to programm some code to solve nonlinear systems of equations with Newton Method. Due to the limited time and space, there are some methods that I cannot expand in this project. However, I would continue my learning in nonlinear systems.