Lecture 36,37,38 - Rootfinding in systems of equations

(A) Theory
(B) Problems
(C) MATLAB Applications

Text: Supplementary notes from Instructor

36.1 Why is it important to be able to find roots to systems of equations?

Up to this point, we have discussed how to find the solution to

- single non-linear equation (Newton Raphson Method)
- systems of linear equations (Linear Algebra)

However, frequently in chemical engineering, we need to find the solution to a system of non-linear equations. This forces us to combine the Newton-Raphson method with Linear Algebra. The technique we will discuss in this section is Multivariate Newton Raphson Method.

36.2 Multivariate Newton-Raphson - Theory

Recall that when we wanted to find the solution to a single non-linear equation of the form

$$f(x) = 0 \tag{31.1}$$

The basis of the Newton-Raphson method lay in the fact that we can approximate the derivative of $f(x)$ numerically.

$$f'(x_1) = \frac{df(x_1)}{dx} \approx \frac{f(x_1) - f(x_2)}{x_1 - x_2} \tag{31.2}$$

leading to the iterative formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{31.5}$$

Now consider the system of n non-linear equations and n unknowns.

$$\begin{aligned}
&f_1(x_1,x_2,x_3,\ldots x_{n-1},x_n)=0\\
&f_2(x_1,x_2,x_3,\ldots x_{n-1},x_n)=0\\
&f_3(x_1,x_2,x_3,\ldots x_{n-1},x_n)=0\\
&\ldots\\
&f_{n-1}(x_1,x_2,x_3,\ldots x_{n-1},x_n)=0\\
&f_n(x_1,x_2,x_3,\ldots x_{n-1},x_n)=0
\end{aligned} \tag{36.1}$$

This is the most general form of the problems that face chemical engineers and encompass linear equations (when all the functions, f, are linear) and single equations (when n = 1).

One technique used to solve this problem is called the multivariate Newton Raphson Method (MNRM). The idea follows from the single-variable case. The basic idea again stems from the fact that the total derivative of a function, f_j, is

$$df_j = \left(\frac{\partial f_j}{\partial x_1}\right)dx_1 + \left(\frac{\partial f_j}{\partial x_2}\right)dx_2 \tag{36.2}$$

for the case of two variables or

$$df_j = \sum_{i=1}^{n}\left(\frac{\partial f_j}{\partial x_i}\right)dx_i \tag{36.3}$$

for the n variable case. We can discretize this expression and write:

$$f_j\left(\{x\}^{(2)}\right) - f_j\left(\{x\}^{(1)}\right) = \sum_{i=1}^{n}\left(\frac{\partial f_j}{\partial x_i}\right)\left(x_i^{(2)} - x_i^{(1)}\right) \tag{36.4}$$

where the j is the index over functions, the i is the index over variables and the superscript in parentheses stands for the iteration. Note that if we have only one variable (n=1), this reduces to the Newton-Raphson method that we have already learned.

Now consider that we have n equations so that j = 1 to n. We have a system of equations which we can write as:

$$\underline{f}\left(\{x\}^{(2)}\right) - \underline{f}\left(\{x\}^{(1)}\right) = \sum_{i=1}^{n}\left(\frac{\partial \underline{f}}{\partial x_i}\right)\left(x_i^{(2)} - x_i^{(1)}\right) \tag{36.5}$$

Just as in the single variable case, we want our next iteration to take us to the root so we assume that $\underline{f}\left(\{x\}^{(2)}\right) = 0$. We can then write this system in matrix notation as:

$$\underline{\underline{J}}^{(k)}\underline{\delta x}^{(k)} = -\underline{R}^{(k)} \qquad (36.6)$$

where $\underline{R}^{(k)}$ is called the residual vector at the k^th^ iteration and is defined as

$$\underline{R}^{(k)} = \underline{f}\left(\{x\}^{(k)}\right) \qquad (36.7)$$

where $\underline{\underline{J}}^{(k)}$ is called the Jacobian matrix at the k^th^ iterationand is defined as

$$\left(\underline{\underline{J}}^{(k)}\right)_{j,i} = \left(\frac{\partial f_j^{(k)}}{\partial x_i^{(k)}}\right) \qquad (36.8)$$

and where

$$\underline{\delta x}^{(k)} = \underline{x}^{(k+1)} - \underline{x}^{(k)} \qquad (36.9)$$

so that the new guess for the $\underline{x}$ is

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{\delta x}^{(k)} \qquad (36.10)$$

The algorithm for solving the multivariate Newton-Raphson follows analogously from the single variable NRM. The steps are as follows:

1. Make an initial guess for $\underline{x}$
2. calculate the Jacobian and the Residual.
3. Solve equation 36.6
4. Calculate new $\underline{x}$ from equation 36.10
5. If the solution has not converged, loop back to step 2.

The multivariate Newton-Raphson Method suffers from the same short-comings as the single-variable Newton-Raphson Method.

(1) You need a good initial guess.
(2) You don't get quadratic convergence until you are close to the solution.
(3) If the partial derivatives are zero, the method blows up. If the partial derivatives are close to zero, the method may not converge.

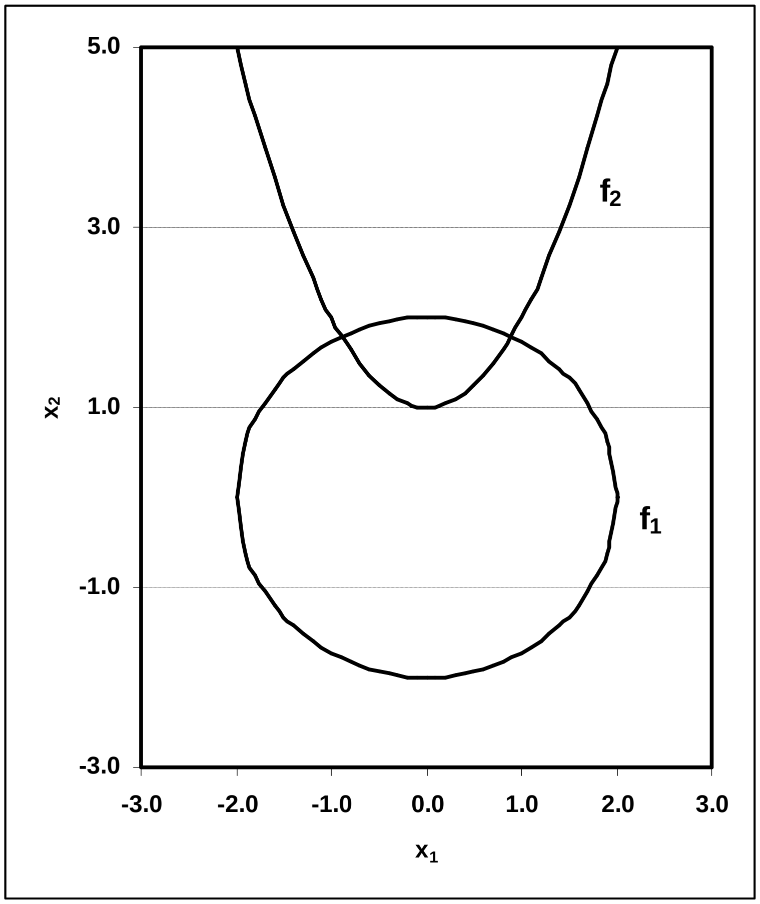
36.3 Multivariate Newton-Raphson - Problems

As with all the numerical techniques we have been using, it is necessary to practice with them before we become proficient with them.

Example One:

$$\begin{aligned} f_1(x_1,x_2) &= (x_1)^2 + (x_2)^2 - 4 = 0 \\ f_2(x_1,x_2) &= (x_1)^2 - (x_2) + 1 = 0 \end{aligned} \qquad (36.11)$$

The x_2 vs $x_1$plot of the following 2 functions is given below. The first function is that of a circle, centered at the origin with radius 2. The second function is a parabola. We see there are two and only two solutions to this systems of equations.



In order to use MNRM, we must first determine the functional form of the partial derivatives

$$(\underline{\underline{J}})_{1,1} = \left(\frac{\partial f_1}{\partial x_1}\right) = 2x_1 \qquad (\underline{\underline{J}})_{1,2} = \left(\frac{\partial f_1}{\partial x_2}\right) = 2x_2$$
$$(\underline{\underline{J}})_{1,2} = \left(\frac{\partial f_2}{\partial x_1}\right) = 2x_1 \qquad (\underline{\underline{J}})_{2,2} = \left(\frac{\partial f_2}{\partial x_2}\right) = -1$$

Then following the algorithm outlined above:

Step One. Make an initial guess. One of the solutions looks to be at $x_1 = 1.0$ and $x_2 = 2.0$.
Step Two. Using that initial guess, calculate the residual and the Jacobian.

$$\underline{\underline{J}}^{(1)} = \begin{bmatrix} 2 & 4 \\ 2 & -1 \end{bmatrix} \text{ and } \underline{R}^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Step Three. Solve $\underline{\underline{J}}^{(k)}\underline{\delta x}^{(k)} = -\underline{R}^{(k)}$ (Using Linear Algebra)

$$\underline{\delta x}^{(1)} = \begin{bmatrix} -0.1 \\ -0.2 \end{bmatrix}$$

Step Four. Calculate new values for $\underline{x}$ via equation (36.10)

$$\underline{x}^{(2)} = \begin{bmatrix} 0.9 \\ 1.8 \end{bmatrix}$$

Step Five. Loop back to Step 2. and repeat until converged.

Here are what further iterations yield

iteration	$\underline{x}$	$\underline{\underline{J}}$	$\underline{R}$	$\underline{\delta x}$
1	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 4 \\ 2 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.1 \\ -0.2 \end{bmatrix}$
2	$\begin{bmatrix} 0.9 \\ 1.8 \end{bmatrix}$	$\begin{bmatrix} 1.8 & 3.6 \\ 1.8 & -1 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ 0.01 \end{bmatrix}$	$\begin{bmatrix} -0.0104 \\ -0.0087 \end{bmatrix}$

3 $$\begin{bmatrix} 0.8896 \\ 17913 \end{bmatrix} \quad \begin{bmatrix} 17792 & 3.5826 \\ 17792 & -10000 \end{bmatrix} \quad \begin{bmatrix} 0.1835e-3 \\ 0.1079e-3 \end{bmatrix} \quad \begin{bmatrix} -0.6991e-4 \\ -0.1650e-4 \end{bmatrix}$$

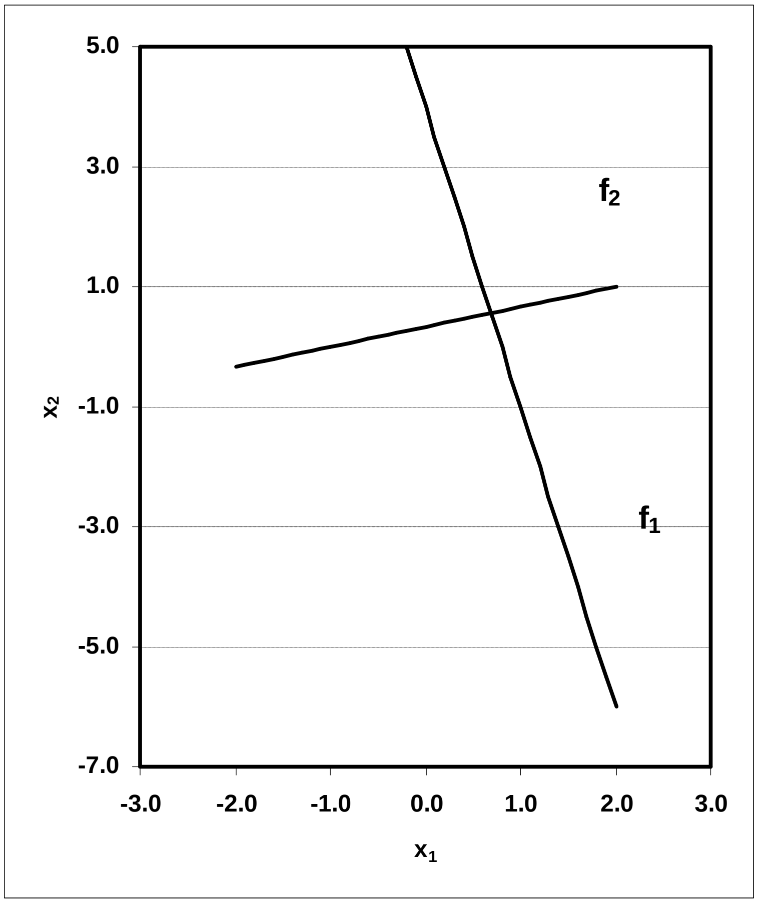
4 $$\begin{bmatrix} 0.8895 \\ 17913 \end{bmatrix} \quad \begin{bmatrix} 17791 & 3.5826 \\ 17791 & -10000 \end{bmatrix} \quad \begin{bmatrix} 0.5159e-8 \\ 0.4887e-8 \end{bmatrix} \quad \begin{bmatrix} -0.2780e-8 \\ -0.0059e-8 \end{bmatrix}$$

The other root in to the problem is located at $\underline{x} = \begin{bmatrix} -0.8895 \\ 1.7913 \end{bmatrix}$ by symmetry.

Example Two.

Linear systems are a subset of non-linear systems. The multivariate Newton-Raphson solve linear systems exactly in one iteration, just as was the case in the single-variable problem.
Consider the system of linear equations:

$$\begin{aligned} f_1(x_1,x_2) &= 5(x_1)+(x_2)-4=0 \\ f_2(x_1,x_2) &= (x_1)-3(x_2)+1=0 \end{aligned} \tag{36.12}$$



In order to use MNRM, we must first determine the functional form of the partial derivatives

$$(\underline{\underline{J}})_{1,1} = \left(\frac{\partial f_1}{\partial x_1}\right) = 5 \qquad (\underline{\underline{J}})_{1,2} = \left(\frac{\partial f_1}{\partial x_2}\right) = 1$$
$$(\underline{\underline{J}})_{1,2} = \left(\frac{\partial f_2}{\partial x_1}\right) = 1 \qquad (\underline{\underline{J}})_{2,2} = \left(\frac{\partial f_2}{\partial x_2}\right) = -3$$

Then following the algorithm outlined above:

Step One. Make an initial guess. One of the solutions looks to be at x_1= 2.0 and x_2 = 2.0 .
Step Two. Using that initial guess, calculate the residual and the Jacobian.

$$\underline{\underline{J}}^{(1)} = \begin{bmatrix} 5 & 1 \\ 1 & -3 \end{bmatrix} \text{ and } \underline{R}^{(1)} = \begin{bmatrix} 8 \\ -3 \end{bmatrix}$$

Step Three. Solve $\underline{\underline{J}}^{(k)}\underline{\delta x}^{(k)} = -\underline{R}^{(k)}$ (Using Linear Algebra)

$$\underline{\delta x}^{(1)} = \begin{bmatrix} \text{-13125} \\ \text{-14375} \end{bmatrix}$$

Step Four. Calculate new values for $\underline{x}$ via equation (36.10)

$$\underline{x}^{(2)} = \begin{bmatrix} 0.6875 \\ 0.5625 \end{bmatrix}$$

A second iteration will show that this is the exact solution.

36.3 Multivariate Newton-Raphson - MATLAB

Very quickly problems like this become too difficult to solve by hand. We rely on software like MATLAB to solve these problems for us. MATLAB does not necessarily use a multivariate Newton-Raphson method to solve a system of non-linear equations but it uses some similar numerical technique. This alternate technique approximates the partial derivatives, so that the user only has to enter the functional form of the functions and not that of the partial derivatives.

On the website, you can download a routine called syseqn.m. This routine will allow you to solve a system of non-linear algebraic equations.

The description for how to use the file can be obtained by opening MATLAB, moving to the directory where you have downloaded the syseqn.m file, and typing

```
help syseqn
```

This yields:

```
syseqn(xo)
  syseqn solves a system of nonlinear algebraic equations.

   The system of equations are stored in the file syseqninput.m
   The initial guesses are given as a vector in xo.
   The solution is written to the screen and to the file "syseqn.out"

   Author:  David Keffer   Date:  October 23, 1998
```

The routine uses the MATLAB instrinsic function fzero to solve the system of equations. At the MATLAB command line, the routine is invoke by typing, for example,

```
syseqn(1)
```

if you had one equation and you wanted your initial guess to be $x_o = 1$. Or, if you had a system of three non-linear equation, you could invoke syseqn.m by typing

```
syseqn([2;4;6])
```

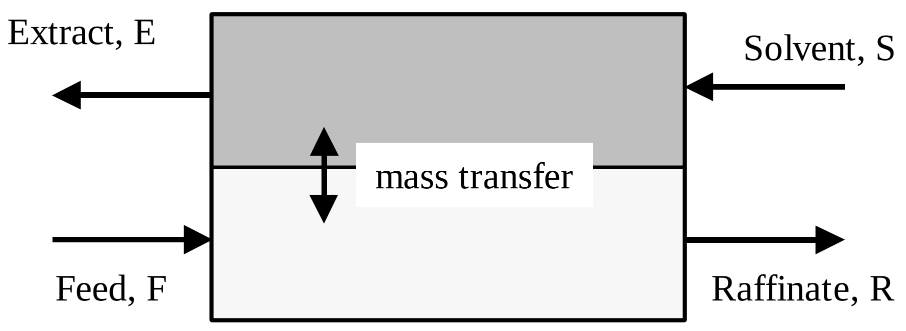
if you had one equation and you wanted your initial guess to be $x_{o,1} = 2$, $x_{o,2} = 4$, $x_{o,3} = 6$.

The algebraic equations are entered in the file syseqninput.m This file can be as simple as

```
function [f] = syseqninput(x0)
f=x^2-1;
```

to solve for the roots of $x^2 - 1 = 0$. The syseqninput.m can also be much more complicated. Two more involved examples of the syseqninput.m for (a) solving a system of mass balances in an extractor, and (b) determining the chemical equilibria compositions and temperature in a vessel where multiple reactions are occuring in a non-adiabatic and non-isothermal environment. In the latter case, you will see that the syseqninput.m file can be very complicated. It doesn't matter how complicated it appears, so long as, at the end of the file, you have evaluated the function of interest.

Example: Consider the following problem from chemical engineering. You want to use an extraction process to remove a contaminant from a feed stream. The process diagram looks like this:



The data you are given is

$$\begin{array}{llll}
F = 100 mol/hr & S = 150 mol/hr & R = ? & E = ? \\
x_{F,b} = 0.1 & x_{S,b} = 0.0001 & x_{R,b} = ? & x_{E,b} = ? \\
x_{F,c} = 0.9 & x_{S,c} = 0.0 & x_{R,c} = ? & x_{E,c} = 0.0 \\
x_{F,f} = 0.0 & x_{S,f} = 0.9999 & x_{R,f} = 0.0 & x_{E,f} = ?
\end{array}$$

You have six unknown variables.

You need six equations to solve these.

You have three mass balances:

$$F + S = R + E$$
$$Fx_{F,c} = Rx_{R,c}$$
$$Sx_{S,f} = Ex_{E,f}$$

You have two constraints on the compositions

$$x_{R,c} + x_{R,b} = 1$$
$$x_{E,c} + x_{E,b} = 1$$

You have one separation ratio:

$$K = 0.68 = \frac{x_{E,b}}{x_{R,b}}$$

These are your six equations. The second, third, and sixth equations are non-linear. You need to use a technique like multivariate Newton-Raphson to solve this problem.

In order to use MNRM to solve this system of non-linear equations, rearrange the equations so that the right hand side is zero

$$f_1 = F + S - R - E = 0$$
$$f_2 = Fx_{F,c} - Rx_{R,c} = 0$$
$$f_3 = Sx_{S,f} - Ex_{E,f} = 0$$
$$f_4 = x_{R,c} + x_{R,b} - 1 = 0$$
$$f_5 = x_{E,c} + x_{E,b} - 1 = 0$$
$$f_6 = 0.68 - \frac{x_{E,b}}{x_{R,b}} = 0$$

MATLAB can solve this problem easily, using the syseqn.m routine provided on the web-site. You don't alter the syseqn.m file. You simply type the equations into the syseqninput.m file and then invoke the syseqn.m routine with reasonable initial guesses.

This routine can then be used to do parameter studies.

1. Look at different compositions in the raffinate as a function of different feed rates or solvent rates or feed or solvent compositions. (In order to change parameters, like F, you only have to change the value of F in extractinput.m.)
2. Look at different cases. Perhaps you want to specify the Raffinate composition and determine the feedrate. (In order to change which functions are variables, you simply make sure that you assign your initial guesses to the appropriate variables at the beginning of the routine.
3. Look at any mass balance system. This code gives the skeleton for solving any system of non-linear equations. All that is required is changing the input file. With a little study of the program, you will be able to modify it and use it in all your classes.

Using this code solves the above example in a fraction of a second:

$F = 100\text{mol}/\text{hr}$	$S = 150\text{mol}/\text{hr}$	$R = 94.74\text{mol}/\text{hr}$	$E = 155.26\text{mol}/\text{hr}$
$x_{F,b} = 0.1$	$x_{S,b} = 0.0001$	$x_{R,b} = 0.05$	$x_{E,b} = 0.034$
$x_{F,c} = 0.9$	$x_{S,c} = 0.0$	$x_{R,c} = 0.95$	$x_{E,c} = 0.0$
$x_{F,f} = 0.0$	$x_{S,f} = 0.9999$	$x_{R,f} = 0.0$	$x_{E,f} = 0.966$