Computational Methods and Modelling

Lecture 3: Dr. Edward McCarthy

Finding Roots of **Multiple** Equations Numerically

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Common Roots of Equation Systems

1. There are occasions where it is necessary to determine the common roots of multiple equations simultaneously in the most efficient way possible (e.g. optimisation)

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

2. The methods we will cover today are designed to solve systems of non-linear equations.
3. We will also cover special methods for efficiently solving higher order polynomials
4. The solution of linear equations is another topic that we will cover later in the course.
5. Let us take the example of the two equations

$$x^2 + xy = 10 \qquad y + 3xy^2 = 57$$

Roots x = 2, y = 3

6. We want to solve for the values of x and y that satisfy both equations at the same time.

Common Roots of Equation Systems

7. The solutions will satisfy the following equations which are re-expressions of the ones just presented.

$$u(x, y) = x^2 + xy - 10 = 0$$

$$v(x, y) = y + 3xy^2 - 57 = 0$$

8. We can used fixed point iteration or Newton-Raphson to solve this problem.
9. Let's start with fixed point iteration.
10. The solution strategy is straightforward. Take initial guess values of both x and y (x_0 and y_0) in this case x = 1.5, and y = 3.5, and generate an improved estimate for x, x_{i+1} by rearranging the first equation in terms of x_i and solving for x_{i+1}

$$x_{i+1} = \frac{10 - x_i^2}{y_i}$$

11. Simultaneously, the second equation is rearranged in terms of y and solved to generate a next estimate for y, y_{i+1}.

$$y_{i+1} = 57 - 3x_i y_i^2$$

Roots of Equation Systems

1. An improved value of x, x_{i+1} is given by the following formula

$$x = \frac{10 - (1.5)^2}{3.5} = 2.21429$$

2. This value is now used with y_0 = 3.5 to generate an improved y-value, y_{i+1} = -24.37..

$$y = 57 - 3(2.21429)(3.5)^2 = -24.37516$$

3. However, this approach doesn't appear to be working as the estimate is diverging radically from the true root of y = 3.

4. One approach to resolve this is to change the form of the original equations to be solved.

5. This is done in order to remove the second powers in both equations and prevent rapid divergence from a solution developing.

6. Consider the following new versions of the original equations

Roots of Equation Systems

7. If we now pursue the same iterative process with the same initial values of x and y, we get a radically different and better performance.
8. On the first guess to calculate x_{i+1} we now get:

$$x = \sqrt{10 - 1.5(3.5)} = 2.17945$$

9. Proceeding to the end of the second iteration of this technique we get:

$$y = \sqrt{\frac{57 - 3.5}{3(2.17945)}} = 2.86051$$

$$x = \sqrt{10 - 2.17945(2.86051)} = 1.94053$$

$$y = \sqrt{\frac{57 - 2.86051}{3(1.94053)}} = 3.04955$$

10. We can see that the values are converging towards the true solutions of x = 2 and y = 3.

Roots of Equation Systems

1. The condition for fixed point iteration to converge on a solution for two non-linear equations is:

$$\left|\frac{\partial u}{\partial x}\right| + \left|\frac{\partial u}{\partial y}\right| < 1 \qquad \left|\frac{\partial v}{\partial x}\right| + \left|\frac{\partial v}{\partial y}\right| < 1$$

2. However, while this technique is relatively straightforward, it is often difficult to implement in practice, particularly where the two conditions above are not satisfied.
3. One would effectively have to test the conditions above, prior to using this technique.
4. It is also too dependent on the form of the equation presented for its solution.
5. **An alternative is to use Newton-Raphson instead.**
6. Firstly, we need to establish the mathematical basis of the method which involves more than just the core Newton-Raphson technique we met in previous lectures where it was applied to find the solution to one equation only.

Roots of Equation Systems

1. The Taylor series expansion for both functions u(x) and v(x) that are functions of x and y are (truncated at first order):

$$u_{i+1} = u_i + (x_{i+1} - x_i)\frac{\partial u_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial u_i}{\partial y}$$
$$v_{i+1} = v_i + (x_{i+1} - x_i)\frac{\partial v_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial v_i}{\partial y}$$

2. When u_{i+1} and v_{i+1} are both zero (i.e., when roots in x and y are found) the following expressions apply:

$$\frac{\partial u_i}{\partial x}x_{i+1} + \frac{\partial u_i}{\partial y}y_{i+1} = -u_i + x_i\frac{\partial u_i}{\partial x} + y_i\frac{\partial u_i}{\partial y}$$
$$\frac{\partial v_i}{\partial x}x_{i+1} + \frac{\partial v_i}{\partial y}y_{i+1} = -v_i + x_i\frac{\partial v_i}{\partial x} + y_i\frac{\partial v_i}{\partial y}$$

3. This leads to the following expressions for x_{i+1} and y_{i+1} respectively.

$$x_{i+1} = x_i - \frac{u_i\frac{\partial v_i}{\partial y} - v_i\frac{\partial u_i}{\partial y}}{\frac{\partial u_i}{\partial x}\frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y}\frac{\partial v_i}{\partial x}} \qquad y_{i+1} = y_i - \frac{v_i\frac{\partial u_i}{\partial x} - u_i\frac{\partial v_i}{\partial x}}{\frac{\partial u_i}{\partial x}\frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y}\frac{\partial v_i}{\partial x}}$$

4. The denominator of both expressions is the determinant of the system (non-zero)

THE UNIVERSITY of EDINBURGH
School of Engineering

Roots of Equation Systems

1. Let us now apply Newton Raphson to our original problem to demonstrate its usefulness
2. We begin by calculating the partial derivatives of the initial function values for the guesses x = 1.5 and y = 3.5

$$\frac{\partial u_0}{\partial x} = 2x + y = 2(1.5) + 3.5 = 6.5 \qquad \frac{\partial u_0}{\partial y} = x = 1.5$$

$$\frac{\partial v_0}{\partial x} = 3y^2 = 3(3.5)^2 = 36.75 \qquad \frac{\partial v_0}{\partial y} = 1 + 6xy = 1 + 6(1.5)(3.5) = 32.5$$

3. The determinant for this system is thus:

$$6.5(32.5) - 1.5(36.75) = 156.125$$

4. The values of the functions at the initial guesses are:

$$u_0 = (1.5)^2 + 1.5(3.5) - 10 = -2.5$$

$$v_0 = 3.5 + 3(1.5)(3.5)^2 - 57 = 1.625$$

Roots of Equation Systems

5. The second set of estimates is calculated using the two formula system

$$x = 1.5 - \frac{-2.5(32.5) - 1.625(1.5)}{156.125} = 2.03603$$

$$y = 3.5 - \frac{1.625(6.5) - (-2.5)(36.75)}{156.125} = 2.84388$$

6. These formulas are adapted versions of the simple Newton Raphson equation for one variable to solve one equation:

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

5. The values above now seem to be converging towards the actual solution, x = 2; y = 3.

6. **Exercise 1: Perform the required number of iterations to prove that this system does converge, and determine the relative error after two further steps. (Hint: Use MATLAB.)**

Roots of Polynomial Equations

1. The equation system we just solved happened to be a two-degree polynomial in both x and y.

2. **The determination of roots for polynomials is a topic in itself** that we will tackle for the remainder of this lecture.

3. Recall that a polynomial takes the following general form

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

4. From mathematics theory we already know that the roots of a polynomial obey the following general rules

- For an n^{th} order equation there are n real or complex roots.
- The roots will not necessarily be distinct (i.e. different from each other)
- For n odd, there is at least one real root.
- For complex roots, they always exist in conjugate pairs, i.e., a +bi; a-bi

Roots of Polynomial Equations

5. Polynomials occur frequently in the context of ordinary differential equations and their solution.

6. In this course, we will focus on the specific root-finding techniques that apply to these polynomials.

7. Firstly, there are a number of practical techniques for manipulating and calculating polynomials in computer code that we will cover first.

8. Consider the following cubic polynomial

$$f_3(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

9. **Question: How many multiplications and additions are required to evaluate this polynomial (assuming we know the values of a_i and x in advance)?**

10. **Answer:** six multiplications and three additions.

11. The number of computations for an n^{th} order polynomial is:

$$n(n+1)/2$$

Roots of Polynomial Equations

1. This number of computations is clearly burdensome and scales considerably with every increasing degree, n.

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

2. One solution to reduce the number of computations is **to employ a nested version of the polynomial**

$$f_3(x) = ((a_3 x + a_2)x + a_1)x + a_0$$

3. In this case, we now only need three multiplications and three additions.

4. This might not seem like much, but radically reduces the number of computations and efficiency of a computation when very high order polynomials are being evaluated.

5. A pseudocode for a nested polynomial evaluation is given by

```
DOFOR j = n, 0, -1
   p = p * x+a(j)
END DO
```

Roots of Polynomial Equations

6. Basically, the first iteration is the evaluation of the first internal bracket term, then the second outside term, followed by all the others until the nests are exhausted.

7. To evaluate both the derivative and the function the following pseudocode would be used:

```
DOFOR j = n, 0, −1
  df = df * x+p
  p = p * x+a(j)
END DO
```

8. **Polynomial deflation** is the process of removing a known root from a calculation as soon as it has been determined to avoid repeat computation

9. Take a fifth order polynomial.

$$f_5(x) = -120 - 46x + 79x^2 - 3x^3 - 7x^4 + x^5$$

10. We can express this in factored form if we have determined the roots in advance

Roots of Polynomial Equations

1. This factored form of the polynomial is

$$f_5(x) = (x+1)(x-4)(x-5)(x+3)(x-2)$$

2. Say we have determined the first root x= -1 by some numerical method, but do not yet know what the other roots are (i.e., we only know the first of these terms)

3. To evaluate the rest of the terms in a simplified manner, it makes sense to divide the overall polynomial by the factor (x+1), and then apply our polynomial to the remaining fourth order polynomial, and so on, until all roots have been determined.

4. The fourth order polynomial after division by (x+3) is:

$$f_4(x) = (x+1)(x-4)(x-5)(x-2) = -40 - 2x + 27x^2 - 10x^3 + x^4$$

5. After determination of the next root, r_2, you could then further divide by (x-r_2) etc.

6. A pseudocode to perform factor division is:

Roots of Polynomial Equations

```
clc;
clear;
n=2;
a=zeros(n+1,1);
a(1,1)=-24;
a(2,1)=2;
a(3,1)=1;
t=-6;
r=a(n+1,1);
a(n+1,1)=0;

for i = n:-1:1;
    s=a(i,1);
    a(i,1)=r;
    r=s+r*t
end
```

7. In the code, a polynomial $f^n(x)$ is divided by a factor (x-r): r is the remainder; n is the degree of the polynomial; a(i) are the coefficients of each quotient determined.

8. **Exercise 2: Use the pseudocode to factorise the following quadratic equation by (x-4):**

$$f(x) = (x-4)(x+6) = x^2 + 2x - 24$$

9. In the code, n = 2; a_0 = -24, a_1 = 2, a_2 = 1; t = 4 , i.e., the negative of the independent term in (x-4)

Roots of Polynomial Equations

1. A similar procedure for factoring an nth order polynomial by a dth order polynomial where d < n.
2. The pseudocode for this is as follows:

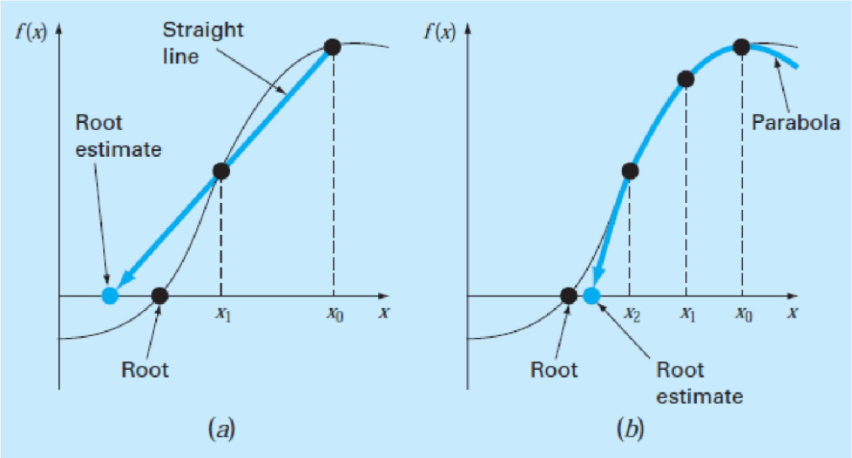
```
SUB poldiv(a, n, d, m, q, r)
DOFOR j = 0, n
  r(j) = a(j)
  q(j) = 0
END DO
DOFOR k = n-m, 0, -1
  q(k+1) = r(m+k) / d(m)
  DOFOR j = m+k-1, k, -1
    r(j) = r(j)-q(k+1) * b(j-k)
  END DO
END DO
DOFOR j = m, n
  r(j) = 0
END DO
n = n-m
DOFOR i = 0, n
  a(i) = q(i+1)
END DO
END SUB
```

Exercise 3:
Apply this to the division of the 5th order polynomial above by the quadratic equation (x+1)(x-4)

Roots of Polynomial Equations

Root Finding Techniques: Muller's Technique

3. Muller's technique is a variant of the Secant Technique, except that **instead of projecting a line to the x-axis, it projects a parabola.**



4. If we take the example of a second degree polynomial (quadratic), the first step is to express the polynomial through any three of its points **about one point, here x_2,** in the following form for each point x_i:

$$f_2(x) = a(x - x_2)^2 + b(x - x_2) + c$$

Roots of Polynomial Equations

1. The three function points along the parabola are:

$$f(x_0) = a(x_0 - x_2)^2 + b(x_0 - x_2) + c$$
$$f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c$$
$$f(x_2) = a(x_2 - x_2)^2 + b(x_2 - x_2) + c$$

2. The three equations allow the determination of the unique set of coefficients a, b and c that satisfy all three equations.

3. The third of these equations allows us to immediately state that $f(x_2) = c$. Thus, we have two remaining equations left:

$$f(x_0) - f(x_2) = a(x_0 - x_2)^2 + b(x_0 - x_2)$$
$$f(x_1) - f(x_2) = a(x_1 - x_2)^2 + b(x_1 - x_2)$$

4. The following terms in the two equations are defined by short-hand terms as follows:

$$h_0 = x_1 - x_0 \qquad h_1 = x_2 - x_1$$

$$\delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \qquad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Roots of Polynomial Equations

5. From these terms, general expressions for the polynomial coefficients can be written as follows:

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0}$$

$$b = ah_1 + \delta_1$$

$$c = f(x_2)$$

Coefficients of a new test polynomial

6. Knowing these coefficients allow us to write the solution to the approximating polynomial as follows.

$$x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

7. In this expression, **x_3 is the next approximation to the root of the function**, **or in certain cases, the root itself**.

8. The relative error of this estimate can be calculated using the following

$$\varepsilon_a = \left| \frac{x_3 - x_2}{x_3} \right| 100\%$$

9. This assumes that the method is delivering an improved estimate in x_3!

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Roots of Polynomial Equations: Muller

1. A numerical example of Muller's method is given for the following function:

$$f(x) = x^3 - 13x - 12$$

2. Let's suppose we guess the initial roots to be x_1 = 4.5, x_2 = 5.5 and x_3 = 5

3. The known roots of this function are -3, -1 and -4, so the above guesses are significantly different from the true values testing this technique properly!

4. Evaluate the function at the guessed points:

$$f(4.5) = 20.625 \qquad f(5.5) = 82.875 \qquad f(5) = 48$$

5. Calculate the h and δ terms:

$$h_0 = 5.5 - 4.5 = 1 \qquad h_1 = 5 - 5.5 = -0.5$$

$$\delta_0 = \frac{82.875 - 20.625}{5.5 - 4.5} = 62.25 \qquad \delta_1 = \frac{48 - 82.875}{5 - 5.5} = 69.75$$

6. Compute the three coefficients of the test polynomial (as yet unknown):

$$a = \frac{69.75 - 62.25}{-0.5 + 1} = 15 \qquad b = 15(-0.5) + 69.75 = 62.25 \qquad c = 48$$

Roots of Polynomial Equations: Muller

7. The next estimate of the first root of the polynomial is given by:

$$x_3 = 5 + \frac{-2(48)}{62.25 + 31.54451} = 3.976487$$

8. Note that the sign of the discriminant, b, was positive, which means that a positive sign is adopted for the square root term in the denominator of the equation above. This determines which root to adopt. (Need this logic test in Python).
9. The error for the estimated root above is:

$$\varepsilon_a = \left|\frac{-1.023513}{3.976487}\right| 100\% = 25.74\%$$

10. The error is significant, so that a new set of root estimates need to chosen to progress further.
11. Here, we replace the original set of variables $[x_0, x_1, x_2]$ with $[x_1, x_2, x_3]$ and repeat.
12. Now the method converges on the root 4.

Roots of Polynomial Equations: Muller

1. The iteration table for the technique through these four iterations is:

$$f(x) = x^3 - 13x - 12$$

i	x_r	ε_a (%)
0	5	
1	3.976487	25.74
2	4.00105	0.6139
3	4	0.0262
4	4	0.0000119

2. To determine the other roots of the equation, one either begins with another initial root estimate well away from x = 4, or to avoid the risk of repeatedly determining x = 4 as a root, one can implement polynomial deflation by dividing the polynomial above by (x-4).

3. Implementation of Muller's technique in an algorithm is done following a pseudocode as per the one opposite: (Note you will need to define your function explicitly or by requesting the user to input it here)