Week 10

Solving Nonlinear Equations

10.1 Motivation

Equations and models used in Engineering can be very broadly categorised as: **linear**, where knowledge of response of system to some external force of a given magnitude implies knowledge of response for any magnitude (e.g. behaviour of a perfectly elastic spring); or **nonlinear** knowledge of response to external force of a single magnitude does not imply knowledge of how system will respond for various magnitudes of external force (non-linear elasticity). Nonlinear equations arise commonly in Engineering problems, e.g.:

- behaviour of structures e.g. nonlinear structural analysis;
- nonlinear oscillations and resonance;
- nonlinear electrical circuits e.g. power losses;
- fluid dynamics; and
- deformations of objects under load.

Nonlinear equations/models are often difficult to solve, so are sometimes approximated via linear equations, however this may not always be valid.

10.1.1 Nonlinear Root-finding

In this module we consider the problem of using numerical techniques to find the **roots** (or **zeros**) of nonlinear equations, $f(x)$. These roots are the solutions to the equation $f(x)=0$. Initially we examine the case where the nonlinear equations are a scalar function of a single independent variable, x. Later, we shall consider the more difficult problem where we have a system of n nonlinear equations with n independent variables, $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$.

Since the roots of a nonlinear function, $f(x)$, cannot in general be expressed in closed form, we must use approximate methods to find them. Usually, iterative techniques are used; these start from an initial approximation of a root, and produce a sequence of approximations

$$\left\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots\right\},$$

which converge toward a root. Convergence to a root is usually possible, provided that the function $f(x)$ is sufficiently smooth and the initial approximation is **close enough** to the root.

10.2 Numerical Considerations

10.2.1 Rate of Convergence

Let

$$\left\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots\right\}$$

be a sequence which converges to a root χ, and let $\varepsilon^{(k)} = \left|x^{(k)} - \chi\right|$. If there exists a number p and a non-zero constant c such that

$$\lim_{k\to\infty} \frac{\varepsilon^{(k+1)}}{\varepsilon^{(k)^p}} = c, \tag{10.1}$$

then p is called the **order of convergence** of the sequence. For $p = 1, 2, 3$ the order is said to be **linear**, **quadratic** and **cubic**, respectively. Linear convergence can be thought of as:

$$\varepsilon^{(k+1)} \approx c\,\varepsilon^{(k)} \qquad c < 1,$$

for sufficiently large k. In fact, values $\varepsilon^{(k)}$, $k \in \{0, 1, \ldots\}$, converge to a geometric sequence as k becomes large.

Note that p need not necessarily be integer; for example, with $p \in (1, 2)$ the convergence is said to be **superlinear**.

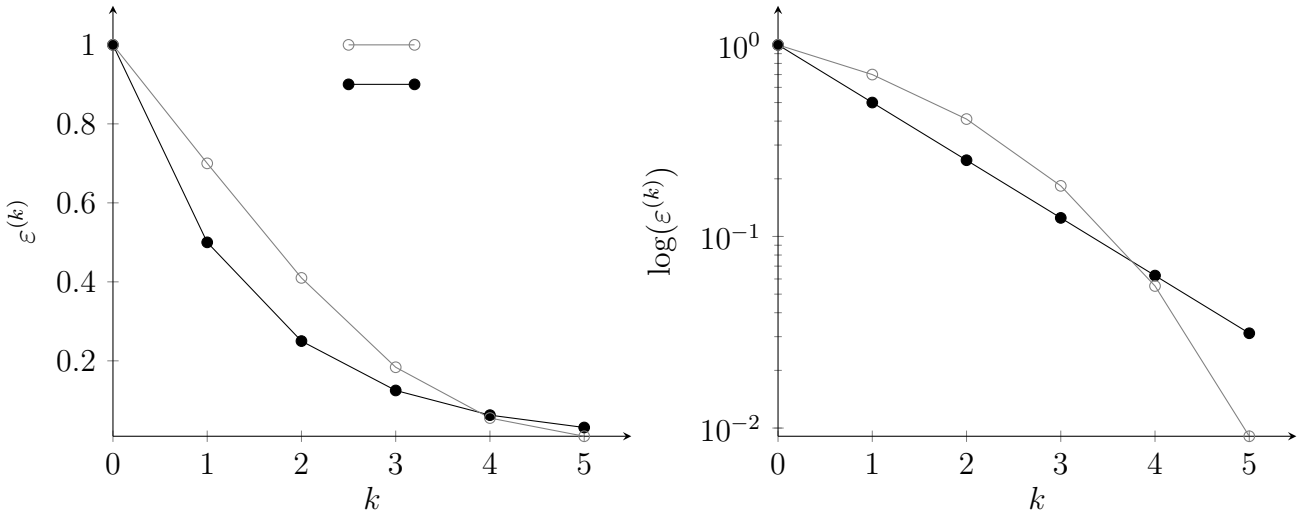


FIGURE 10.1

10.2.2 Termination Criteria

Since numerical computations are performed using floating point arithmetic, there will always be a finite precision to which a function can be evaluated. Let δ denote the **limiting accuracy** of

the nonlinear function near a root. This limiting accuracy of the function imposes a limit on the accuracy, ϵ, of the root. The accuracy to which the root may be found is given by:

$$\epsilon = \frac{\delta}{|f'(\chi)|}. \tag{10.2}$$

This is the best error bound for **any** root finding method. Note that ϵ is large when the first derivative of the function at the root, $|f'(\chi)|$, is small. In this case the problem of finding the root is ill-conditioned. This is shown graphically in Figure 10.2.

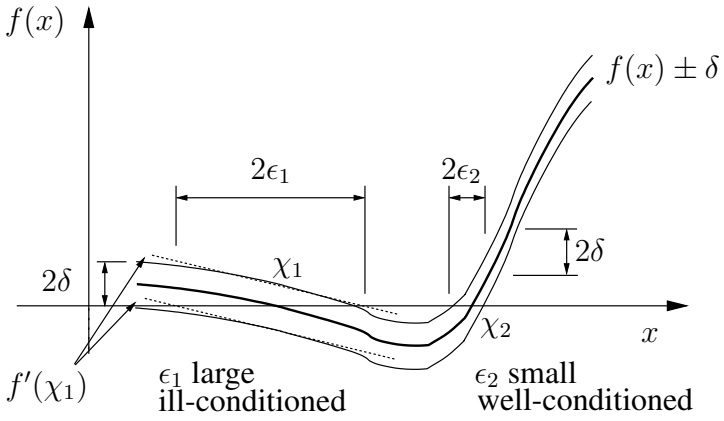


FIGURE 10.2

When the sequence

$$\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots\}$$

converges to the root, χ, then the differences $\left|x^{(k)} - x^{(k-1)}\right|$ will decrease until $\left|x^{(k)} - \chi\right| = \varepsilon^{(k)} \approx \epsilon$. With further iterations, rounding errors will dominate and the differences will vary irregularly.

There are different termination conditions that can be used, depending on the required accuracy. The first method is known as the **residual test**, with the algorithm terminated when:

$$|f(x^{(k)})| < \Delta_1, \tag{10.3}$$

where Δ_1 is the maximum function value that you will accept for the value $x^{(k)}$ to be considered a good enough estimate of a root.

The second method is called the **uniform test**, with the algorithm terminated when

$$\frac{\left|x^{(k)} - x^{(k-1)}\right|}{1 + |x^{(k)}|} < \Delta_2, \tag{10.4}$$

where Δ_2 is some small tolerance value. The condition (10.4) is able to test the **relative offset** when $\left|x^{(k)}\right|$ is much larger than 1, and the **absolute offset** when $\left|x^{(k)}\right|$ is much smaller than 1.

However, if you require the maximum accuracy for the root estimate, you should employ the following test in conjunction with either (10.3) or (10.4).

$$|x^{(k)} - x^{(k-1)}| \leq |x^{(k+1)} - x^{(k)}|$$

Explain what the above test does. How can this improve the accuracy of the root estimate?

In practice, computation time is a larger concern than perfect accuracy, so a condition that the number of iterations not exceed some user-defined limit is often employed as well.

10.3 Root-finding Methods for Nonlinear Scalar Functions

A continuous one-dimensional function $f(x)$ may have any number of roots (including no roots at all). However, if we evaluate the function at two points a and b, and find $f(a) \times f(b) < 0$, then this is sufficient for us to know that there is at least one root in the interval $x \in (a, b)$. In this case we say that a root is **bracketed** on the interval (a, b).

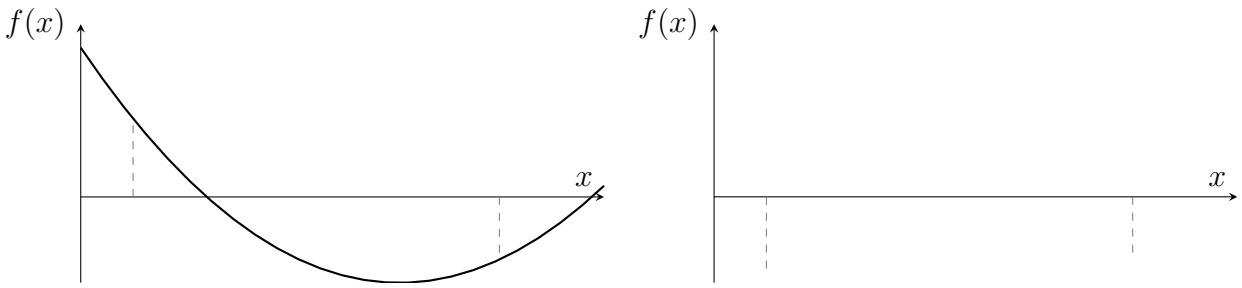


FIGURE 10.3

10.3.1 Bisection Method

The **bisection method** makes use of the above result by assuming that we have two initial approximations of the root, $x^{(0)}$ and $x^{(1)}$ such that $f(x^{(0)})$ and $f(x^{(1)})$ have opposite signs; the algorithm is outlined below.

Let

$$x^{(2)} = x^{(1)} - \frac{x^{(1)} - x^{(0)}}{2} = \frac{x^{(0)} + x^{(2)}}{2} \tag{10.5}$$

be the mid-point of the interval $\left(x^{(0)}, x^{(1)}\right)$. Three mutually exclusive possibilities exist:

- if $f\left(x^{(2)}\right) = 0$ then a root has been found;
- if $f\left(x^{(2)}\right)$ has the same sign as $f\left(x^{(0)}\right)$ then a root is in the interval $\left(\mathsf{x(2),x(1)}\right)$;
- if $f\left(x^{(2)}\right)$ has the same sign as $f\left(x^{(1)}\right)$ then a root is in the interval $\left(\mathsf{x(0),x(2)}\right)$.

In the last two cases, the size of the interval bracketing the root has decreased by a factor of two. The next iteration is performed by evaluating the function at the mid-point of the new interval. After k iterations, the size of the interval bracketing the root has decreased to:

$$\mathsf{width\ of\ bracket} = \frac{x^{(1)} - x^{(0)}}{2^{\mathsf{k}}}.$$

The process is shown graphically in Figure 10.4. Given the assumptions (continuity of $f(x)$, and a bracketed root), the bisection method is guaranteed to converge to a root. If the initial interval brackets more than one root, then the bisection method will converge to one of them.

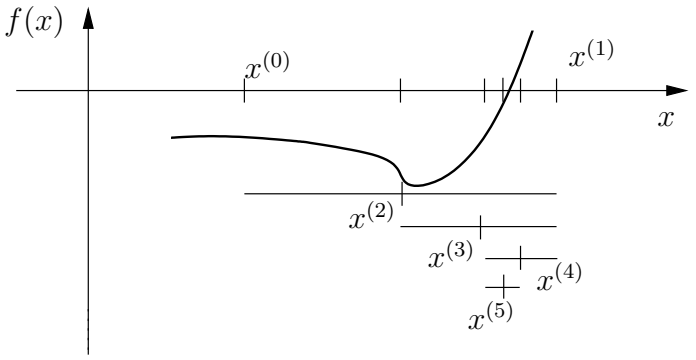


FIGURE 10.4

Since the interval size is reduced by a factor of two at each iteration, it is simple to calculate in advance the number of iterations, k, required to achieve a given tolerance, ϵ_0, in the solution:

$$\epsilon_0 = \frac{x^{(1)} - x^{(0)}}{2^k}$$
$$\Rightarrow \quad 2^k = \frac{x^{(1)} - x^{(0)}}{\epsilon_0}$$

.

The bisection method has a relatively slow convergence. What are the (approximate) values of p and c from equation (10.1) for this method?

10.3.2 Secant Method

The bisection method uses no information about the function values, $f(x)$, apart from whether they are positive or negative at certain values of x. Suppose that

$$\left|f\left(x^{(k)}\right)\right| < \left|f\left(x^{(k-1)}\right)\right|,$$

then we would expect the root to lie closer to $x^{(k)}$ than $x^{(k-1)}$. Instead of choosing the new estimate to lie at the midpoint of the current interval, as is the case with the bisection method, the **secant** method chooses the x-intercept of the secant line to the curve, the line through $\left(x^{(k-1)}, f\left(x^{(k-1)}\right)\right)$ and $\left(x^{(k)}, f\left(x^{(k)}\right)\right)$. This places the new estimate closer to the endpoint for which $f(x)$ has the smallest absolute value. The new estimate is:

$$x^{(k+1)} = x^{(k)} - (x^{(k)} - x^{(k-1)})\frac{f\left(x^{(k)}\right)}{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}. \tag{10.6}$$

Note that the secant method requires two initial function evaluations, but only one new function evaluation is made at each iteration. The secant method is shown graphically in Figure 10.5.

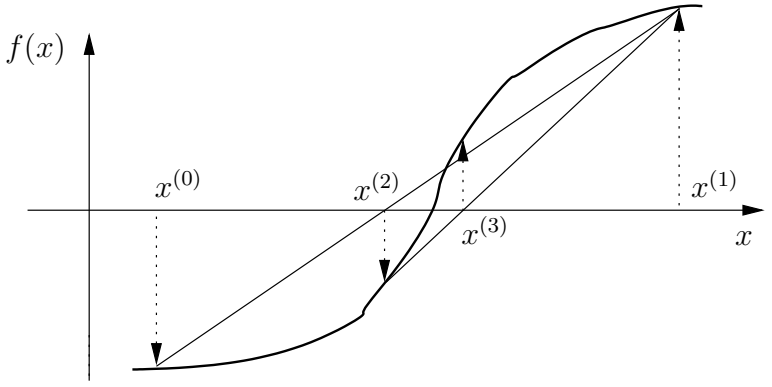


FIGURE 10.5

The secant method does not have the root bracketing property of the bisection method since the new estimate, $x^{(k+1)}$, of the root need not lie within the bounds defined by $x^{(k-1)}$ and $x^{(k)}$. As a consequence, the secant method does not always converge, but when it does, it usually does so faster than the bisection method. It can be shown that the order of convergence of the secant method is:

$$p = \frac{1+\sqrt{5}}{2} \approx 1.618.$$

10.3.3 Regula Falsi

Regula falsi is a variant of the secant method. The difference between the secant method and regula falsi lies in the choice of points used to form the secant. While the secant method uses the two most recent function evaluations to form the secant line through $\left(x^{(k-1)}, f\left(x^{(k-1)}\right)\right)$ and $\left(x^{(k)}, f\left(x^{(k)}\right)\right)$, regula falsi forms the secant line through the most recent points that bracket the root. The regula falsi steps are shown graphically in Figure 10.6. In this example, the point $x^{(0)}$ remains active for many steps.

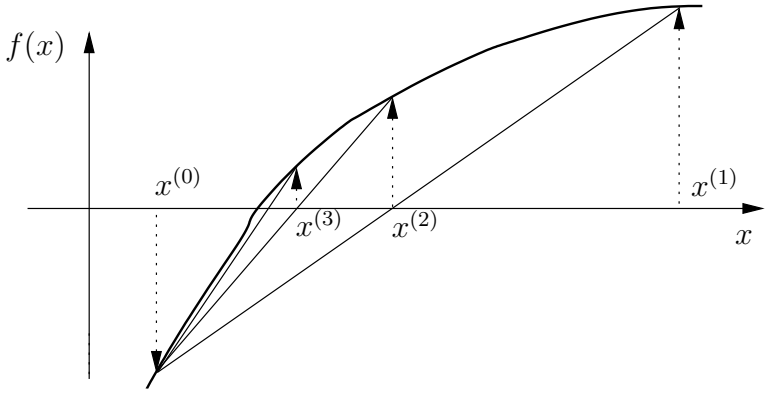


FIGURE 10.6

The advantage of regula falsi is that, like the bisection method, it is always convergent. However, like the bisection method, it has only linear convergence. Examples where the regula falsi method is slow to converge are not hard to find. One example is shown in Figure 10.7.

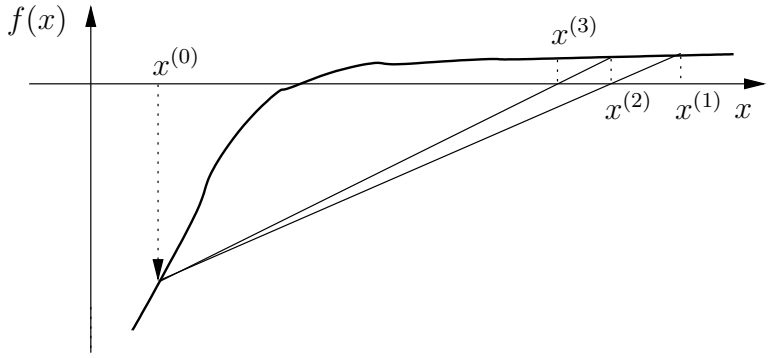


FIGURE 10.7

Perform 2 iterations of (a) the secant method, and (b) the regula falsi method on the function $f(x) = x^3 - x^2 - 1$, starting with $x^{(0)} = 2$, $x^{(1)} = 1$.

SECANT METHOD

k	$x^{(k-1)}$	$x^{(k)}$	$f\left(x^{(k-1)}\right)$	$f\left(x^{(k)}\right)$	$x^{(k+1)}$
1	2	1	3	-1	1.25
2	1	1.25	-1	-0.609375	1.353375

REGULA FALSI

k	x^L	x^R	$f\left(x^L\right)$	$f\left(x^R\right)$	$x^{(k+1)}$	$f(x^{(k+1)})$
1	1	2	-1	3	1.25	0.609375
2	1.25	2	-0.609375	3	1.3766	–

10.3.4 Newton's Method

The methods discussed so far have required only function values in order to compute the new estimate of the root. Newton's method requires that both the function value and the first derivative be evaluated. Geometrically, Newton's method proceeds by extending the tangent line to the function at the current point until it crosses the x-axis. The new estimate of the root is taken as the abscissa of the zero crossing. Newton's method is defined by the following iteration:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}. \tag{10.7}$$

The update is shown graphically in Figure 10.8.

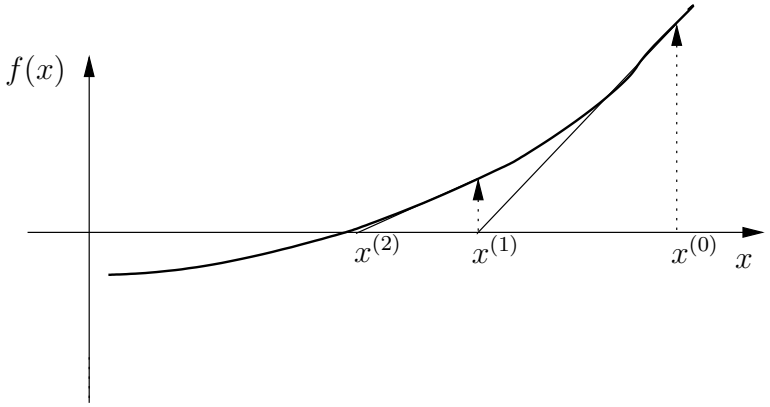


FIGURE 10.8

Newton's method may be derived from the Taylor series expansion of the function:

$$f\left(x + \Delta x\right) = f\left(x\right) + f'(x)\Delta x + f''(x)\frac{(\Delta x)^2}{2} + \ldots \tag{10.8}$$

For a smooth function and small values of Δx, the function is approximated well by the first two terms. Thus

$$f\left(x + \Delta x\right) = 0 \Rightarrow \Delta x = \text{-f(x)/f'(x)}$$

Far from a root, the higher-order terms are significant and Newton's method can give highly inaccurate corrections. In such cases the Newton iterations **may never converge** to a root. In order to achieve convergence, the starting value must be reasonably close to a root. An example of divergence using Newton's method is given in Figure 10.9.

Newton's method exhibits quadratic convergence. Thus, near a root, the number of significant digits doubles with each iteration. The strong convergence makes Newton's method attractive in cases where the derivatives can be evaluated efficiently and the derivative is continuous and non-zero in the neighbourhood of the root (as is the case with multiple roots).

Whether the secant method should be used in preference to Newton's method depends upon the relative work required to compute the first derivative of the function. If the work required to evaluate the first derivative is greater than 0.44 times the work required to evaluate the function, then use the secant method, otherwise use Newton's method.

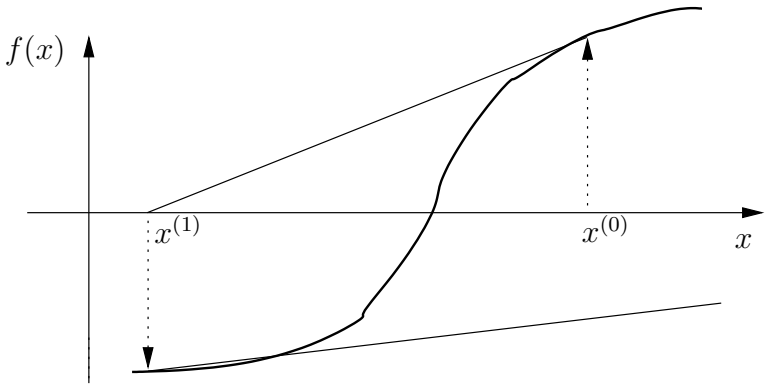


FIGURE 10.9

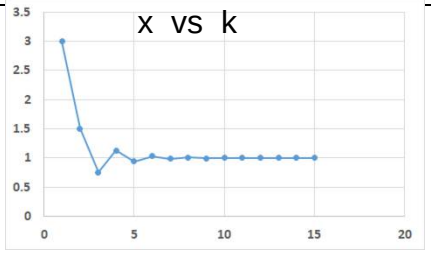
10.3.5 Examples

In the following examples, the bisection, secant, regula falsi and Newton's methods are applied to find the root of the nonlinear function $f(x) = x^2 - 1$ between $[0, 3]$. x_L and x_R are the left and right bracketing values and x_k is the new value determined at each iteration. ε is the distance from the true solution. The bisection method is terminated when condition (10.4) is satisfied for $\Delta_2 = 1 \times 10^{-4}$. The remaining methods determine the solution to the same level of accuracy as the bisection method.

The examples show the linear convergence rate of the Bisection and regula falsi methods, the better than linear convergence rate of the secant method and the quadratic convergence rate of Newton's method. It is also seen that although Newton's method converges in 5 iterations, 5 function and 5 derivative evaluations, to give a total of 10 evaluations, are required. This contrasts with the secant method, where 9 function evaluations are made, for the 8 iterations.

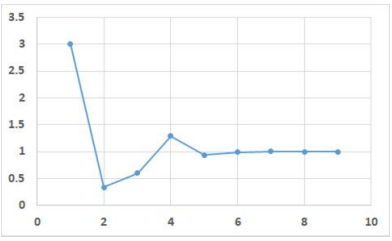
Bisection Method: $f(x) = x^2 - 1$

iteration	k	x_k	x_L	x_R	$f(x_L)$	$f(x_R)$	$f(x_k)$	$\lvert x_k - x_{k-1}\rvert$	$\lvert x_k - x_{k-1}\rvert/(1+\lvert x_k\rvert)$	ε_k
0	0	**0.000000**	–	–	–	–	-1.000000	–	–	1.000000
0	1	**3.000000**	0.000000	3.000000	-1.000000	8.000000	8.000000	3.000000	0.750000	2.000000
1	2	**1.500000**	0.000000	1.500000	-1.000000	1.250000	1.250000	1.500000	0.600000	0.500000
2	3	**0.750000**	0.750000	1.500000	-0.437500	1.250000	-0.437500	0.750000	0.428571	0.250000
3	4	**1.125000**	0.750000	1.125000	-0.437500	0.265625	0.265625	0.375000	0.176471	0.125000
4	5	**0.937500**	0.937500	1.125000	-0.121094	0.265625	-0.121094	0.187500	0.096774	0.062500
5	6	**1.031250**	0.937500	1.031250	-0.121094	0.063477	0.063477	0.093750	0.046154	0.031250
6	7	**0.984375**	0.984375	1.031250	-0.031006	0.063477	-0.031006	0.046875	0.023622	0.015625
7	8	**1.007813**	0.984375	1.007813	-0.031006	0.015686	0.015686	0.023438	0.011673	0.007813
8	9	**0.996094**	0.996094	1.007813	-0.007797	0.015686	-0.007797	0.011719	0.005871	0.003906
9	10	**1.001953**	0.996094	1.001953	-0.007797	0.003910	0.003910	0.005859	0.002927	0.001953
10	11	**0.999023**	0.999023	1.001953	-0.001952	0.003910	-0.001952	0.002930	0.001466	0.000977
11	12	**1.000488**	0.999023	1.000488	-0.001952	0.000977	0.000977	0.001465	0.000732	0.000488
12	13	**0.999756**	0.999756	1.000488	-0.000488	0.000977	-0.000488	0.000732	0.000366	0.000244
13	14	**1.000122**	0.999756	1.000122	-0.000488	0.000244	0.000244	0.000366	0.000183	0.000122
14	15	**0.999939**	–	–	–	–	–	0.000183	0.000092	0.000061



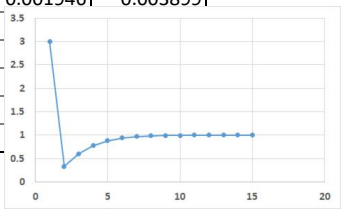
Secent Method: $f(x) = x^2 - 1$

iteration	k	x_k	x_{k-1}	$f(x_k)$	$f(x_{k-1})$	$\lvert x_k - x_{k-1}\rvert$	$\lvert x_k - x_{k-1}\rvert/(1+\lvert x_k\rvert)$	ε_k
0	0	**0.000000**	–	-1.000000	–	–	–	1.000000
0	1	**3.000000**	0.000000	8.000000	-1.000000	3.000000	0.750000	2.000000
1	2	**0.333333**	3.000000	-0.888889	8.000000	2.666667	2.000000	0.666667
2	3	**0.600000**	0.333333	-0.640000	-0.888889	0.266667	0.166667	0.400000
3	4	**1.285714**	0.600000	0.653061	-0.640000	0.685714	0.300000	0.285714
4	5	**0.939394**	1.285714	-0.117539	0.653061	0.346320	0.178571	0.060606
5	6	**0.992218**	0.939394	-0.015504	-0.117539	0.052824	0.026515	0.007782
6	7	**1.000244**	0.992218	0.000488	-0.015504	0.008026	0.004013	0.000244
7	8	**0.999999**	1.000244	-0.000002	0.000488	0.000245	0.000123	0.000001
8	9	**1.000000**	–	–	–	0.000001	0.000000	0.000000



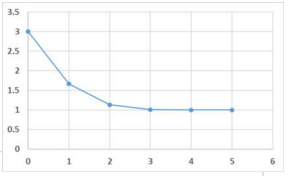
Regula Falsi Method: $f(x) = x^2 - 1$

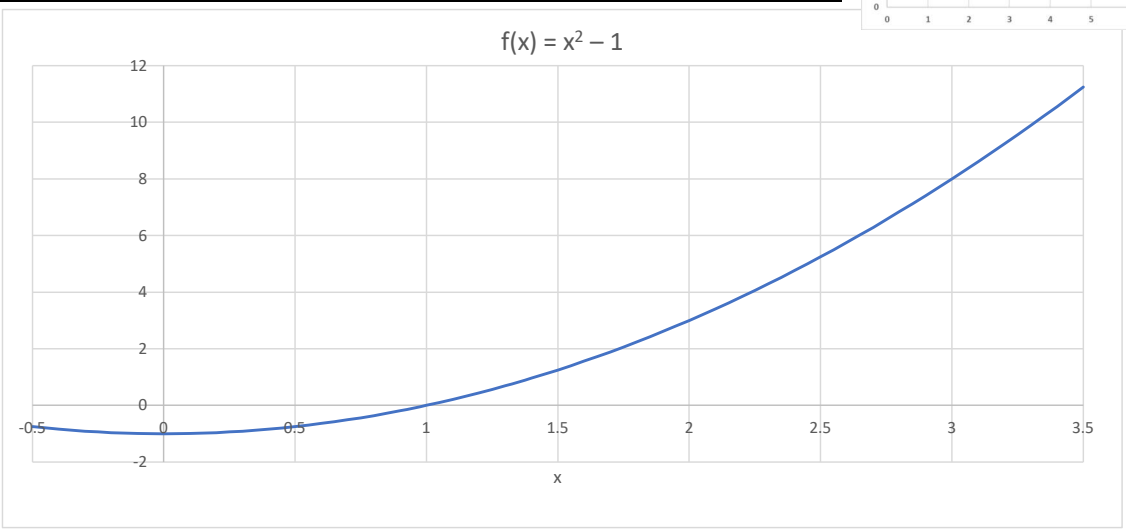
iteration	k	x_k	x_L	x_R	$f(x_L)$	$f(x_R)$	$f(x_k)$	$\lvert x_k - x_{k-1}\rvert$	$\lvert x_k - x_{k-1}\rvert/(1+\lvert x_k\rvert)$	ε_k
0	0	**0.000000**	–	–	–	–	-1.000000	–	–	1.000000
0	1	**3.000000**	0.000000	3.000000	-1.000000	8.000000	8.000000	3.000000	0.750000	2.000000
1	2	**0.333333**	0.333333	3.000000	-0.888889	8.000000	-0.888889	2.666667	2.000000	0.666667
2	3	**0.600000**	0.600000	3.000000	-0.640000	8.000000	-0.640000	0.266667	0.166667	0.400000
3	4	**0.777778**	0.777778	3.000000	-0.395062	8.000000	-0.395062	0.177778	0.100000	0.222222
4	5	**0.882353**	0.882353	3.000000	-0.221453	8.000000	-0.221453	0.104575	0.055556	0.117647
5	6	**0.939394**	0.939394	3.000000	-0.117539	8.000000	-0.117539	0.057041	0.029412	0.060606
6	7	**0.969231**	0.969231	3.000000	-0.060592	8.000000	-0.060592	0.029837	0.015152	0.030769
7	8	**0.984496**	0.984496	3.000000	-0.030767	8.000000	-0.030767	0.015265	0.007692	0.015504
8	9	**0.992218**	0.992218	3.000000	-0.015504	8.000000	-0.015504	0.007722	0.003876	0.007782
9	10	**0.996101**	0.996101	3.000000	-0.007782	8.000000	-0.007782	0.003883	0.001946	0.003899
10	11	**0.998049**	0.998049	3.000000	-0.003899	8.000000	-0.003899	0.001947		
11	12	**0.999024**	0.999024	3.000000	-0.001951	8.000000	-0.001951	0.000975		
12	13	**0.999512**	0.999512	3.000000	-0.000976	8.000000	-0.000976	0.000488		
13	14	**0.999756**	0.999756	3.000000	-0.000488	8.000000	-0.000488	0.000244		
14	15	**0.999878**	–	–	–	–	–	0.000122		



Newton's Method: $f(x) = x^2 - 1$; $f'(x) = 2x$

iteration	k	x_k	$f(x_k)$	$f'(x_k)$	$\lvert x_k - x_{k-1}\rvert$	$\lvert x_k - x_{k-1}\rvert/(1+\lvert x_k\rvert)$	ε_k
0	0	**3.000000**	8.000000	6.000000	–	–	2.000000
1	1	**1.666667**	1.777778	3.333333	1.333333	0.500000	2.000000
2	2	**1.133333**	0.284444	2.266667	0.533333	0.250000	0.133333
3	3	**1.007843**	0.015748	2.015686	0.125490	0.062500	0.007843
4	4	**1.000031**	0.000061	2.000061	0.007813	0.003906	0.000031
5	5	**1.000000**	–	–	0.000031	0.000015	0.000000





10.3.6 Combining Methods

Newton's method and the secant method do not always converge to the desired root, they can converge to a different root outside the initial bracket, or can just diverge away from the desired root. In these cases we would want to change the method to the regula falsi method, until we are close enough to the root that Newton's or the secant method would start to converge.

It is also possible to circumvent the poor global convergence properties of Newton's method by combining it with the bisection method. This hybrid method uses a bisection step whenever the Newton method takes the solution outside the bisection bracket. Global convergence is thus assured while retaining quadratic convergence near the root.

We may also want to combine methods when using the regula falsi method. The regula falsi method can have a convergence rate that is slower than bisection, if the function is one-sided around the root.

10.4 Laguerre's Method for Polynomial Equations

In the previous sections we have investigated methods for finding the roots of general functions. These techniques may sometimes work for finding roots of polynomials, but often problems arise because of the existence of multiple roots or because we are interested in finding complex roots. Better methods are available for nonlinear functions with these types of roots. One such technique is Laguerre's method.

Consider the n^{th} order polynomial:

$$P_n(x) = (x - x_1)(x - x_2)\dots(x - x_n).$$

Taking the logarithm of the absolute value of both sides gives:

$$\ln|P_n(x)| = \ln|x - x_1| + \ln|x - x_2| + \ldots + \ln|x - x_n|\,.$$

Let:

$$\begin{aligned} G &= \frac{d\ln|P_n(x)|}{dx} \\ &= \frac{1}{x - x_1} + \frac{1}{x - x_2} + \ldots + \frac{1}{x - x_n} && (10.9) \\ &= \frac{P_n'(x)}{P_n(x)} && (10.10) \end{aligned}$$

and

$$\begin{aligned} H &= -\frac{d^2\ln|P_n(x)|}{dx^2} \\ &= \frac{1}{(x - x_1)^2} + \frac{1}{(x - x_2)^2} + \ldots + \frac{1}{(x - x_n)^2} && (10.11) \\ &= \left(\frac{P_n'(x)}{P_n(x)}\right)^2 - \frac{P_n''(x)}{P_n(x)}. && (10.12) \end{aligned}$$

If we make the assumption that the root x_1 is located a distance α from our current estimate, $x^{(k)}$, and all other roots are located at a distance β, then:

$$x^{(k)} - x_1 = \alpha \tag{10.13}$$

$$x^{(k)} - x_i = \beta, \quad \forall i \in \{2, 3, \ldots, n\}. \tag{10.14}$$

Equations (10.9) and (10.11) can now be expressed as:

$$G = \frac{1}{\alpha} + \frac{n-1}{\beta} \tag{10.15}$$

$$H = \frac{1}{\alpha^2} + \frac{n-1}{\beta^2}, \tag{10.16}$$

where $G = \frac{P_n'(x)}{P_n(x)}$ and $H = \left(\frac{P_n'(x)}{P_n(x)}\right)^2 - \frac{P_n''(x)}{P_n(x)}$.

Solving for α gives:

$$\alpha = \frac{n}{G \pm \sqrt{(n-1)\left(nH - G^2\right)}} \tag{10.17}$$

where the sign should be taken to yield the largest magnitude for the denominator. The new estimate of the root is obtained from the previous estimate, using the update:

$$x^{(k+1)} = x^{(k)} - \alpha. \tag{10.18}$$

In general, α will be complex.

Laguerre's method requires that $P_n\left(x^{(k)}\right)$, $P_n'\left(x^{(k)}\right)$ and $P_n''\left(x^{(k)}\right)$ be computed at each step. The method is cubic in its convergence for real or complex simple roots. Laguerre's method will converge to all types of roots of polynomials, real, complex, single or multiple. It requires complex arithmetic even when converging to a real root. However, for polynomials with all real roots, it is guaranteed to converge to a root from any starting point.

After a root, x_1, of the n^{th} order polynomial, $P_n(x)$, has been found, the polynomial should be divided by the quotient, $(x - x_1)$, to yield an $(n-1)$ order polynomial. This process is called **deflation**. It saves computation when estimating further roots and ensures that subsequent iterations do not converge to roots already found.

10.4.1 Example

Following is a simple example that illustrates the use of Laguerre's method. Consider the quadratic polynomial:

$$P_2(x) = x^2 - 4x + 3, \qquad P_2'(x) = 2x - 4, \qquad P_2''(x) = 2.$$

1. Let the initial estimate of the first root be:

$$x_1^{(0)} = 0.5$$

then

$$P_2(0.5) = 1.25, \qquad P_2'(0.5) = -3, \qquad P_2''(x) = 2,$$

$$G = \frac{-3}{1.25} = -2.4$$
$$H = G^2 - \frac{2}{1.25} = 4.16$$
$$\alpha = \frac{2}{G \pm \sqrt{(2-1)(2H - G^2)}} = -0.5 \text{ (choosing } -4 \text{ as the largest denominator)}$$

The updated estimate for the root is then:

$$x^{(1)} = x^{(0)} - \alpha = 0.5 - (-0.5) = 1.$$

Now we recompute:

$$P_2(1) = 0, \qquad P_2'(1) = -2, \qquad P_2''(1) = 2.$$

Since $P_2(1) = 0$, we have found a root.

2. The second root is found by dividing P_2 by $(x_1 - 1)$; the trivial result is that $x_2 = 3$.

10.4.2 Horner's Method

The polynomial $P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$ should not be calculated by evaluating the powers of x individually. It is much more efficient to calculate the polynomial using Horner's method by noting that $P_n(x) = a_0 + x(a_1 + x(a_2 + \ldots + x a_n))$. Using this approach the polynomial value and the first and second derivatives may be evaluated efficiently using the following algorithm:

$p \leftarrow a_n$;

$p' \leftarrow 0.0$;

$p'' \leftarrow 0.0$;

for i in $n-1 \ldots 0$ loop

$p'' \leftarrow x \times p'' + p'$;
$p' \leftarrow x \times p' + p$;
$p \leftarrow x \times p + a_i$;

end loop;

$p'' \leftarrow 2 \times p''$;

If the polynomial and its derivatives are evaluated using the individual powers of x, $(n^2+3n)/2$ operations are required for the value, $(n^2+n-2)/2$ for the first derivative and $(n^2-n-2)/2$ for the second derivative, i.e. a total of $(3n^2+3n-4)/2$ operations. Horner's method requires $6n+1$ operations to evaluate the polynomial and both derivatives and thus requires fewer operations when $n > 3$. If just the function evaluation is required, then Horner's method will require fewer operations for all n.

Use Horner's method to evaluate

$$P_3(x) = 1 + 2x - 3x^2 + x^3$$

and its derivatives, by filling in the table below.

P''			P'				P				
x^0		x^1	x^0		x^1	x^2	x^0		x^1	x^2	x^3
0	+	0	0	+	0 +	0	1	+	0 +	0 +	0
0	+	0	1	+	0 +	0	-3	+	1 +	0 +	0
1	+	0	-3	+	1 +	0	2	−	3 +	1 +	0
-3	+	3	2	−	6 +	3	1	+	2 −	3 +	1
-6	+	$6x$	2	−	$6x$ +	$3x^2$	1	+	$2x$ −	$3x^2$ +	x^3

10.4.3 Deflation

Dividing a polynomial of order n by a factor $(x - x_1)$ may be performed using the following algorithm:

$r \leftarrow a_n$;

$a_n \leftarrow 0.0$;

for i in $n-1 \ldots 0$ loop

$q \leftarrow a_i$;

$a_i \leftarrow r$;

$r \leftarrow x_1 \times r + q$;

end loop;

The coefficients of the new polynomial are stored in the array a_i, and the remainder in r.

Suppose we know that one root of the polynomial:

$$P(x) = x^3 + x^2 - 5x - 2$$

is $x_1 = 2$. Use long division to find a quadratic polynomial with the remaining two roots, and compare the operations with those in the algorithm above.

10.5 Systems of Nonlinear Equations

Consider a general system of n nonlinear equations with n unknowns:

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad \forall i \in \{1, 2, \ldots, n\}.$$

This can be written in vector form as:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

Newton's method can be generalised to n dimensions by examining Taylor's series in n dimensions; for the i^{th} equation we have:

$$f_i\left(\boldsymbol{x}^{(k+1)}\right) = f_i\left(\boldsymbol{x}^{(k)}\right) + \sum_{j=1}^{n} f'_{ij}\left(\boldsymbol{x}^{(k)}\right)\left(x_j^{(k+1)} - x_j^{(k)}\right) + \ldots$$

where $f'_{ij}\left(\boldsymbol{x}^{(k)}\right)$ is the element in the i^{th} row and j^{th} column of the $n \times n$ matrix called the **Jacobian**; specifically each element is defined as follows:

$$f'_{ij}(\boldsymbol{x}^{(k)}) = \left.\frac{\partial f_i(\boldsymbol{x})}{\partial x_j}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(k)}} \quad \forall i \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, n\}.$$

Setting $f_i\left(\boldsymbol{x}^{(k+1)}\right) = 0$ and truncating the Taylor series gives Newton's formula in n dimensions:

$$\sum_{j=1}^{n} \underbrace{f'_{ij}\left(\boldsymbol{x}^{(k)}\right)}_{\text{Matrix}}\left(x_j^{(k+1)} - x_j^{(k)}\right) = -f_i\left(\boldsymbol{x}^{(k)}\right), \quad \forall i \in \{1, 2, \ldots, n\},$$

or

$$\sum_{j=1}^{n} f'_{ij}(\boldsymbol{x}^{(k)})\delta_j^{(k+1)} = -f_i\left(\boldsymbol{x}^{(k)}\right), \quad \forall i \in \{1, 2, \ldots, n\}, \tag{10.19}$$

where $\delta_j^{(k+1)} = x_j^{(k+1)} - x_j^{(k)}$ is the update for the j^{th} variable. Equation (10.19) is a set of n linear equations with n unknowns, $\delta_j^{(k+1)}$. If the Jacobian is non-singular, then the system of linear equations:

$$\boldsymbol{f}'(\boldsymbol{x}^{(k)})\boldsymbol{\delta}^{(k+1)} = -\boldsymbol{f}(\boldsymbol{x}^{(k)})$$

can be solved to provide the Newton update:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{\delta}^{(k+1)}.$$

Question: under what circumstances might the Jacobian be singular or ill-conditioned?

Each step of Newton's method requires the solution of a set of linear equations. For small n the set of linear equations may be solved using LU decomposition. For large n, alternative iterative methods may be required. As with the one-dimensional version, Newton's method converges quadratically if the initial estimate, $\boldsymbol{x}^{(0)}$, is sufficiently close to a root. Newton's method in multiple dimensions suffers from the same global convergence problems as its one-dimensional counterpart.

Quasi-Newton method

If the gradients are not provided, or cannot be computed analytically they can be approximately calculated, the method is called a **quasi-Newton** method. One way to approximately form the gradients is to calculate $f_i(\boldsymbol{x} - \boldsymbol{\Delta x_j})$ and $f_i(\boldsymbol{x} + \boldsymbol{\Delta x_j})$ and then form the finite difference approximation:

$$f'_{ij}(\boldsymbol{x}) \approx \frac{f_i(\boldsymbol{x} + \boldsymbol{\Delta x_j}) - f_i(\boldsymbol{x} - \boldsymbol{\Delta x_j})}{2\Delta x_j}, \quad \forall i \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, n\}. \tag{10.20}$$

The vector $\boldsymbol{\Delta x_j}$ is a vector of zeros, except for a small perturbation in the j^{th} position.

10.5.1 Example 1

Consider finding x and y such that the following are true:

$$f_1(x, y) = x^2 - 2x + y^2 + 2y - 2xy + 1 = 0 \tag{10.21}$$
$$f_2(x, y) = x^2 + 2x + y^2 + 2y + 2xy + 1 = 0 \tag{10.22}$$

This is a contrived problem, but it serves to illustrate the application of Newton method's in multiple dimensions. Solving for the simultaneous roots of these equations is equivalent to finding the roots of the single factored equation:

$$f(x, y) = (x - y - 1)^2 + (x + y + 1)^2 \tag{10.23}$$

Question: Why is this valid?

Inspection shows that this equation has the root $(0, -1)$.

The first step in applying Newton's method is to determine the form of the Jacobian and the right-hand side function vector for calculating the update vector for each iteration. These are:

$$\begin{bmatrix} \dfrac{\partial f_1(x^{(k)}, y^{(k)})}{\partial x} & \dfrac{\partial f_1(x^{(k)}, y^{(k)})}{\partial y} \\ \dfrac{\partial f_2(x^{(k)}, y^{(k)})}{\partial x} & \dfrac{\partial f_2(x^{(k)}, y^{(k)})}{\partial y} \end{bmatrix} \begin{bmatrix} \delta_x^{(k+1)} \\ \delta_y^{(k+1)} \end{bmatrix} = \begin{bmatrix} -f_1(x^{(k)}, y^{(k)}) \\ -f_2(x^{(k)}, y^{(k)}) \end{bmatrix} \tag{10.24}$$

$$\begin{bmatrix} 2(x - y - 1) & 2(y - x + 1) \\ 2(x + y + 1) & 2(x + y + 1) \end{bmatrix} \begin{bmatrix} \delta_x^{(k+1)} \\ \delta_y^{(k+1)} \end{bmatrix} = \begin{bmatrix} -x^2 + 2x - y^2 - 2y + 2xy - 1 \\ -x^2 - 2x - y^2 - 2y - 2xy - 1 \end{bmatrix} \tag{10.25}$$

The algorithm for solving this problem is:

$k \leftarrow 0$;

$\epsilon \leftarrow M > \Delta$;

specify starting point: $(x^{(k)}, y^{(k)})$;

while: $\epsilon > \Delta$ and $k < N$

$\boldsymbol{\delta}^{(k+1)} \leftarrow -\boldsymbol{f}'^{-1}(\boldsymbol{x}^{(k)})\boldsymbol{f}(\boldsymbol{x}^{(k)})$; (solve using LU factorisation)
$\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \boldsymbol{\delta}^{(k+1)}$;
$\epsilon \leftarrow ||\boldsymbol{f}(\boldsymbol{x}^{(k+1)})||_\infty$;
$k \leftarrow k + 1$;

end loop;

Where N is some specified maximum number of iterations and Δ is some specified convergence tolerance for the magnitude of the residuals.

Figure 10.10 shows the convergent trajectories for the Newton's method as a quiver plot.

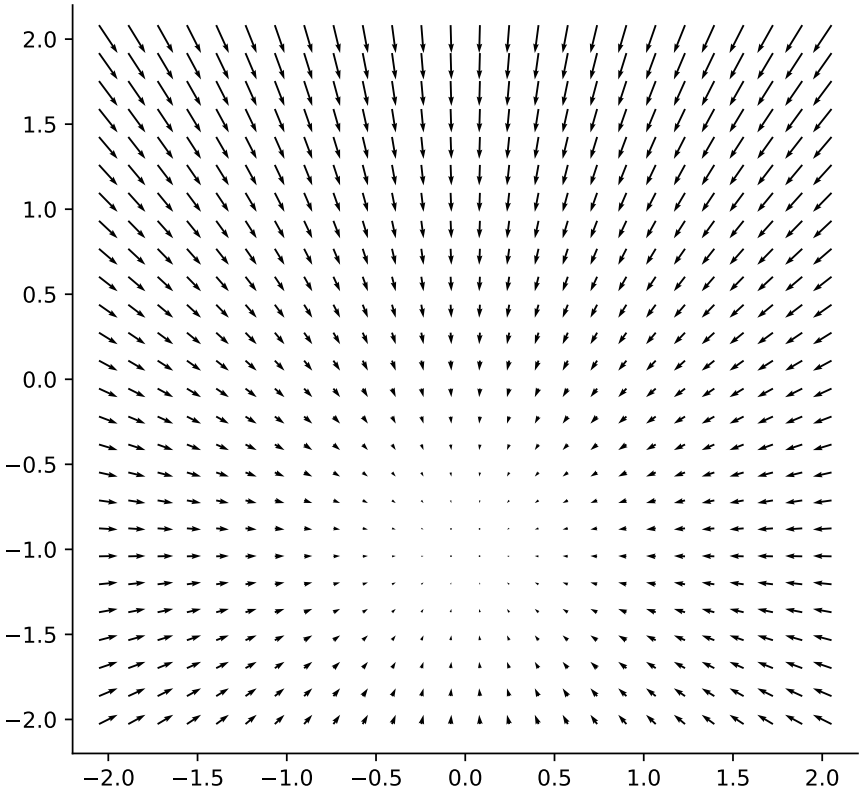


FIGURE 10.10: Quiver plot showing the convergence of Newton's method.

10.5.2 Example 2

$$
\begin{aligned}
f_1(x,y) &= x^2 - y = 0 \\
f_2(x,y) &= x^2 + y^2 - 2 = 0.
\end{aligned}
$$

$$
\boldsymbol{f}'(x^{(k)}, y^{(k)}) = \begin{bmatrix} 2x^{(k)} & -1 \\ 2x^{(k)} & 2y^{(k)} \end{bmatrix}
$$

$$
\Rightarrow \quad \begin{bmatrix} 2x^{(k)} & -1 \\ 2x^{(k)} & 2y^{(k)} \end{bmatrix} \begin{bmatrix} \delta_x^{(k+1)} \\ \delta_y^{(k+1)} \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}
$$

$$
\Rightarrow \quad \begin{bmatrix} \delta_x^{(k+1)} \\ \delta_y^{(k+1)} \end{bmatrix} = \begin{bmatrix} \rule{3.5cm}{0.4pt} \\ \\ \rule{1.8cm}{0.4pt} \end{bmatrix}
$$

Question: under what conditions is $\boldsymbol{f}'(x,y)$ singular?

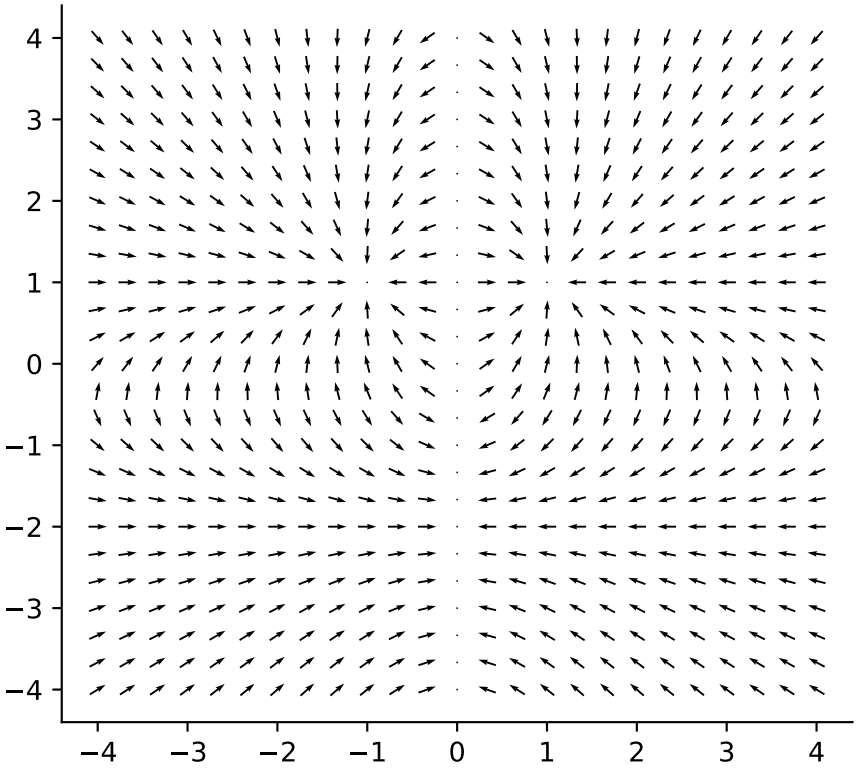


FIGURE 10.11: Normalised quiver plot showing the convergence of the algorithm.

A note on convergence

As stated in the one-dimensional case, Newton's method typically exhibits quadratic convergence. However, if the solution occurs with a gradient of 0 the method only exhibits linear convergence. This issue extends to the multidimensional case, meaning that some roots may be approached more rapidly than others.

10.5.3 Stability and Sensitivity

Newton's method is not always **stable**, this lack of stability typically comes about when the Jacobian is ill-conditioned (leading to very large step sizes). It is also possible that even if the initial estimate for the solution is near a true solution, the algorithm could still diverge.

A related concept is the **sensitivity** of the method to the initial estimate. For (systems of) equations with multiple solutions, typically you would want algorithm to converge to the closest root to the initial estimate. If there is a high sensitivity, this means that small changes in the initial estimate can often change the solution (that the algorithm converges to) considerably.

One common method to improve both the stability and sensitivity of Newton's method is to introduce a scaling (damping) factor (< 1) on the update. This can be written as follows:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{\delta}^{(k+1)},$$

where

$$\alpha^{(k+1)} = \frac{1}{1 + \beta||\delta^{(k+1)}||_2}.$$

β is chosen based on the scale of the problem. For example, if $\beta = 9$, and $||\delta^{(k+1)}||_2 = 1 \Rightarrow \alpha^{(k+1)} = 0.1$, and as $\delta^{(k+1)} \to 0$, $\alpha^{(k+1)} \to 1$. This maintains quadratic (or at least superlinear) convergence near the root, but prevents large jumps in the initial iterations. Below we demonstrate this concept on a single equation with complex roots: $f(z) = z^5 - 1 = 0$, with $\beta = 0$, and $\beta = 9$.

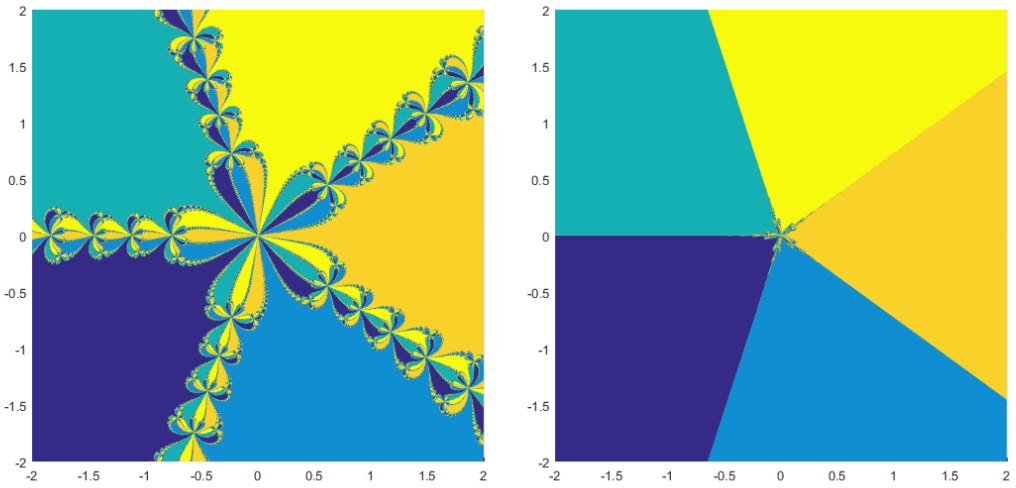


FIGURE 10.12: Shading at a point indicates the converged root, from that starting location.

Root finding methods – things to ponder

What is a root?

Why find roots?

Can there be multiple roots?

What is a bracketed interval?

Can a nonlinear equation have no root at all?

Broadly speaking, what is the ‘secant’ of a curve?

When does a secant become a tangent?

What is a jacobian?

When will Jacobian be singular or ill-conditioned?

What is the difference between Secant and Regula Falsi method?