0.1 Newton's Method, Continued

- Newton's algorithm is widely used because, it is more rapidly convergent than any of the methods discussed so far. **Quadratically convergent**
- The error of each step approaches a constant K times the square of the error of the previous step.
- The number of decimal places of accuracy nearly doubles at each iteration.
- When Newton's method is applied to $f(x) = 3x + sinx - e^x = 0$, if we begin with $x_0 = 0.0$:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0.0 - \frac{-1.0}{3.0} = 0.33333$$

$$x_2 = 0.36017$$

$$x_3 = 0.3604217$$

- After three iterations, the root is correct to seven digits (.3604217029603 2440136932951583028); convergence is much more rapid than any previous method.
- In fact, the error after an iteration is about one-third of the square of the previous error.
- There is the need for two functions evaluations at each step, $f(x_n)$ and $f'(x_n)$ and we must obtain the derivative function at the start.
- If a difficult problem requires many iterations to converge, the number of function evaluations with Newton's method may be many more than with linear iteration methods.
- Because Newton's method always uses two per iteration whereas the others take only one.
- **An algorithm for the Newton's method :**

To determine a root of $f(x) = 0$, given x_0 reasonably close to the root,
Compute $f(x_0), f'(x_0)$
If $(f(x_0) \neq 0)$ And $(f'(x_0) \neq 0)$ Then
Repeat
Set $x_1 = x_0$
Set $x_0 = x_0 - \frac{f(x_0)}{f'(x_0)}$
Until $(|x_1 - x_0| < tolerance\ value1)$ Or If $|f(x_0)| < tolerance\ value2)$
End If.

- The method may converge to a root different from the expected one or diverge if the starting value is not close enough to the root.
- In some cases Newton's method will not converge (Fig. 1).

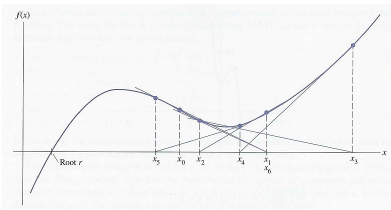

Figure 1: Graphical illustration of the case that Newton's Method will not converge.

- Starting with x_0, one never reaches the root r because $x_6 = x_1$ and we are in an endless loop.
- Observe also that if we should ever reach the minimum or maximum of the curve, we will fly off to infinity.
- **Example**: Apply Newton's method to $x - x^{1/3} - 2 = 0$.
(**m-file: demoNewton.m**. $>> demoNewton(3)$
- **Example**: A general implementation of Newton's method.
(**m-files: newton.m**),(**fx3n.m**).
$>> newton('fx3n', 3, 5e-16, 5e-16, 1)$

0.2 Muller's Method

- Most of the root-finding methods that we have considered so far have approximated the function in the neighbourhood of the root by a *straight line*.
- *Muller's method* is based on approximating the function in the neighbourhood of the root by a *quadratic polynomial*.

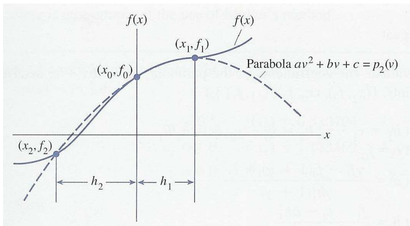

Figure 2: Parabola $a\nu^2 + b\nu + c = p_2(\nu)$

- A second-degree polynomial is made to fit *three points* near a root, at x_0, x_1, x_2 with x_0 between x_1, and x_2.
- The proper *zero of this quadratic*, using the quadratic formula, is used as the improved estimate of the root.
- A quadratic equation that fits through three points in the vicinity of a root, in the form $a\nu^2 + b\nu + c$. (See Fig. 2)
- Transform axes to pass through the middle point, let
  - $\nu = x - x_0$,
  - $h_1 = x_1 - x_0$
  - $h_2 = x_0 - x_2$.

  $$\nu = 0 \Longrightarrow \; a(0)^2 + b(0) + c = f_0$$
  $$\nu = h_1 \Longrightarrow \; ah_1^2 + bh_1 + c = f_1$$
  $$\nu = -h_2 \Longrightarrow \; ah_2^2 - bh_2 + c = f_2$$

  We evaluate the coefficients by evaluating $p_2(\nu)$ at the three points:
- From the first equation, $c = f_0$.

- Letting $h_2/h_1 = \gamma$, we can solve the other two equations for a, and b.

$$a = \frac{\gamma f_1 - f_0(1+\gamma) + f_2}{\gamma h_1^2(1+\gamma)}, \quad b = \frac{f_1 - f_0 - a h_1^2}{h_1}$$

- After computing a, b, and c, we solve for the root of $a\nu^2 + b\nu + c$ by the quadratic formula

$$\nu_{1,2} = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}},$$

$$\nu = x - x_0,$$

$$root = x_0 - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

See Figs. 3-4 that an example is given

Find a root between 0 and 1 of the same transcendental function as before: $f(x) = 3x + \sin(x) - e^x$. Let

$$x_0 = 0.5, \quad f(x_0) = 0.330704 \quad h_1 = 0.5,$$
$$x_1 = 1.0, \quad f(x_1) = 1.123489 \quad h_2 = 0.5,$$
$$x = 0.0, \quad f(x_2) = -1 \quad \gamma = 1.0.$$

Then

$$a = \frac{(1.0)(1.123189) - 0.330704(2.0) + (-1)}{1.0(0.5)^2(2.0)} = -1.07644,$$

$$b = \frac{1.123189 - 0.330704 - (-1.07644)(0.5)^2}{0.5} = 2.12319,$$

$$c = 0.330704,$$

and

$$\text{root} = 0.5 - \frac{2(0.330704)}{2.12319 + \sqrt{(2.12319)^2 - 4(-1.07644)(0.330704)}}.$$
$$= 0.354914.$$

For the next iteration, we have

$$x_0 = 0.354914, \quad f(x_0) = -0.0138066 \quad h_1 = 0.145086,$$
$$x_1 = 0.5, \quad f(x_1) = 0.330704 \quad h_2 = 0.354914,$$
$$x_2 = 0, \quad f(x_2) = -1 \quad \gamma = 2.44623.$$

Figure 3: An example of the use of Muller's method.

- *Experience shows that Muller's method converges at a rate that is similar to that for Newton's method.*

- It does not require the evaluation of derivatives, however, and (after we have obtained the starting values) needs only one function evaluation per iteration.

Then

$$a = \frac{(2.44623)(0.330704) - (-0.0138066)(3.44623) + (-1)}{2.44623(0.145086)^2(3.44623)} = -0.808314,$$

$$b = \frac{0.330704 - (-0.0138066) - (-0.808314)(0.145086)^2}{0.145086} = 2.49180,$$

$$c = -0.0138066,$$

$$\text{root} = 0.354914 - \frac{2(-0.0138066)}{2.49180 + \sqrt{(2.49180)^2 - 4(-0.808314)(-0.0138066)}}$$

$$= 0.360465.$$

After a third iteration, we get 0.3604217 as the value for the root, which is identical to that from Newton's method after three iterations.

Figure 4: Cont. An example of the use of Muller's method.

An algorithm for Muller's method :

Given the points x_2, x_0, x_1 in increasing value,
Evaluate the corresponding function values: f_2, f_0, f_1.
Repeat
(Evaluate the coefficients of the parabola, $a\nu^2+b\nu+c$, determined by the three points.
$(x_2, f_2), (x_0, f_0), (x_l, f_1)$.)
Set $h_l = x_1 - x_0; h_2 = x_0 - x_2; \gamma = h_2/h_1$.
Set $c = f_0$
Set $a = \frac{\gamma f_1 - f_0(1+\gamma)+f_2}{\gamma h_1^2(1+\gamma)}$
Set $b = \frac{f_1 - f_0 - a h_1^2}{h_1}$
(Next, compute the roots of the polynomial.)
Set $root = x_0 - \frac{2c}{b \pm \sqrt{b^2-4ac}}$
Choose root, x_r, closest to x_0 by making the denominator as large as possible; i.e. if
$b > 0$, choose plus; otherwise, choose minus.
If $x_r > x_0$,
Then rearrange to:x_0, x_1, and the root
Else rearrange to: x_0, x_2, and the root
End If.
(In either case, reset subscripts so that x_0, is in the middle.)
Until $|f(x_r)| < Ftol$

0.3 Fixed-point Iteration; $x = g(x)$ Method

- Rearrange $f(x)$ into an equivalent form $x = g(x)$,
- This can be done in several ways.
  - Observe that if $f(r) = 0$, where r is a root of $f(x)$, it follows that $r = g(r)$.

- Whenever we have $r = g(r)$, r is said to be a *fixed point* for the function g.

- The iterative form:

$$x_{n+1} = g(x_n); \quad n = 0, 1, 2, 3, \ldots$$

  converges to the fixed point r, a root of $f(x)$.

- **Example**: $f(x) = x^2 - 2x - 3 = 0$

- Suppose we rearrange to give this equivalent form:

$$x = g_1(x) = \sqrt{2x + 3}$$

$$\begin{array}{lll} x_0 = 4 & \rightarrow & x_1 = \sqrt{11} = 3.31662 \\ x_2 = \sqrt{9.63325} = 3.10375 & \rightarrow & x_3 = 3.03439 \\ x_4 = 3.01144 & \rightarrow & \underline{\underline{x_5 = 3.00381}} \end{array}$$

- If we start with $x = 4$ and iterate with the fixed-point algorithm,

- The values are *converging on the root* at $x = 3$.

0.3.1 Other Rearrangements

- Another rearrangement of $f(x)$; Let us start the iterations again with $x_0 = 4$. Successive values then are:

$$x = g_2(x) = \frac{3}{(x - 2)}$$

$$\begin{array}{lllll} x_0 = 4 & \rightarrow & x_1 = 1.5 & \rightarrow \\ x_2 = -6 & \rightarrow & x_3 = -0.375 & \rightarrow \\ x_4 = -1.263158 & \rightarrow & x_5 = -0.919355 & \rightarrow \\ x_5 = -0.919355 & \rightarrow & x_6 = -1.02762 & \rightarrow \\ x_7 = -0.990876 & \rightarrow & \underline{\underline{x_8 = -1.00305}} \end{array}$$

- It seems that we now converge to the other root, at $x = -1$.

- Consider a third rearrangement; starting again with $x_0 = 4$, we get

$$x = g_3(x) = \frac{(x^2 - 3)}{2}$$

$$\begin{array}{llll} x_0 = 4 & \rightarrow & x_1 = 6.5 & \rightarrow \\ x_2 = 19.625 & \rightarrow & \underline{\underline{x_3 = 191.070}} \end{array}$$

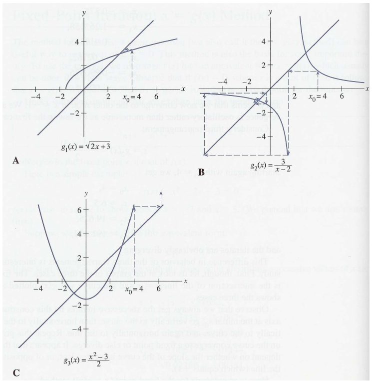

Figure 5: The fixed point of $x = g(x)$ is the intersection of the line $y = x$ and the curve $y = g(x)$ plotted against x. Where A:$x = g_1(x) = \sqrt{2x+3}$. B:$x = g_2(x) = \frac{3}{(x-2)}$. C: $x = g_3(x) = \frac{(x^2-3)}{2}$.

- The iterations are obviously diverging.
- The fixed point of $x = g(x)$ is the intersection of the line $y = x$ and the curve $y = g(x)$ plotted against x.

Figure 5 shows the three cases.

- Start on the x-axis at the initial x_0, go vertically to the curve, then horizontally to the line $y = x$, then vertically to the curve, and again horizontally to the line.
- Repeat this process until the points on the curve converge to a fixed point or else diverge.
- *The method may converge to a root different from the expected one, or it may diverge.*
- *Different rearrangements will converge at different rates.*
- **Iteration algorithm with the form** $x = g(x)$

Table 1: The order of convergence for the iteration algorithm with the different forms of $x = g(x)$.

	If $g(x) = \sqrt{2x+3}$		If $g(x) = 3/(x-2)$	
Iteration	Error	Ratio	Error	Ratio
1	0.31662	0.31662	2.50000	0.50000
2	0.10375	0.32767	−5.00000	−2.00000
3	0.03439	0.33143	0.62500	−0.12500
4	0.01144	0.33270	−0.26316	−0.42105
5	0.00381	0.33312	0.08065	−0.30645
6			−0.02762	−0.34254
7			0.00912	−0.33029
8			−0.00305	−0.33435

To determine a root of $f(x) = 0$, given a value x_1 reasonably close to the root
Rearrange the equation to an equivalent form $x = g(x)$
Repeat
Set $x_2 = x_l$
Set $x_l = g(x_1)$
Until $|x_1 - x_2| <$ *tolerance value*

0.3.2 Order of Convergence

- The fixed-point method converges at a linear rate;
- it is said to be *linearly convergent*, meaning that the error at each successive iteration is a constant fraction of the previous error.
- If we tabulate the errors after each step in getting the roots of the polynomial and its ratio to the previous error,
- we find that the magnitudes of the ratios to be levelling out at 0.3333. (See Table 1)
- **Example**: Comparing Muller's and Fixed-point Iteration methods (**m-files: mainmulfix.m**, **muller.m**, **fixedpoint.m**)

0.4 Multiple Roots

- A function can have more than one root of the same value. See Fig. 6left.
- $f(x) = (x-1)(e^{(x-1)} - 1)$ has a double root at $x = 1$, as seen in Fig. 6right.

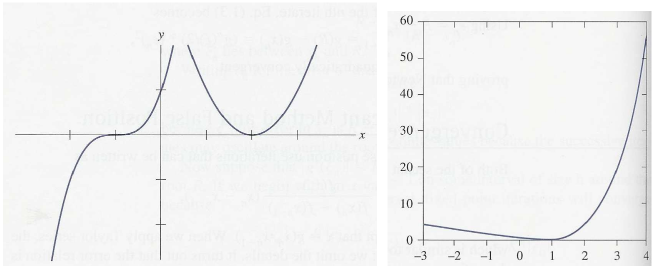

Figure 6: Left: The curve on the left has a triple root at $x = -1$ [the function is $(x+1)^3$]. The curve on the right has a double root at $x = 2$ [the function is $(x-2)^2$].Right: Plot of $(x-1)(e^{(x-1)} - 1)$.

Table 2: Left: Errors when finding a double root. Right: Successive errors with Newton's method for $f(x) = (x+1)^3 = 0$ (Triple root).

Iteration	Error	Ratio
1	0.3679	
2	0.1666	0.453
3	0.0798	0.479
4	0.0391	0.490
5	0.0193	0.494
6	0.0096	0.497
7	0.0048	0.500
8	0.0024	0.500

Iteration	Error	Iteration	Error
0	0.5	6	0.0439
1	0.3333	7	0.0293
2	0.2222	8	0.0195
3	0.1482	9	0.0130
4	0.0988	10	0.00867
5	0.0658		

- The methods we have described do not work well for multiple roots.
- For example, Newton's method is only linearly convergent at a double root.
- Table 2left gives the errors of successive iterates (Newton's method is applied to a double root) and the convergence is clearly linear with ratio of errors is $\frac{1}{2}$.
- When Newton's method is applied to a triple root, convergence is still linear, as seen in Table 2right. The ratio of errors is larger, about $\frac{2}{3}$.

```
>> x = linspace( -4, 4, 100 );plot(x,x.^3+3*x.^2+3*x+1); grid on
>> x= linspace( -4, 4, 100 );plot(x,x.*exp(x-1)-x-exp(x-1)+1); grid on
>> x = linspace( 0, 4, 1500 );plot(x,x.^2-4*x+4); grid on
```

0.5 The fzero function

- The **MATLAB** *fzero* function is a hybrid of bisection, the secant method, and interpolation.
- Care is taken to *avoid unnecessary calculations and to minimize the effects of roundoff.*

```
>> xb=brackPlot('fx3',0,5);
>> fzero('fx3',xb)
ans =     3.5214
options=optimset('Display','iter');
r=fzero('(x+1)^3',[-10 10],options)
```

0.6 Nonlinear Systems

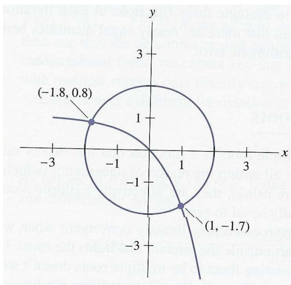

Figure 7: A pair of equations.

- A pair of equations:
  $x^2 + y^2 = 4$
  $e^x + y = 1$
- Graphically, the solution to this system is represented by the intersections of the circle $x^2 + y^2 = 4$ with the curve $y = 1 - e^x$ (see Fig. 7)
- Newton's method can be applied to systems as well as to a single nonlinear equation. We begin with the forms

$$f(x, y) = 0,$$
$$g(x, y) = 0$$

- Let

$$x = r, y = s$$

be a **root**.

- Expand both functions as a Taylor series about the point (x_i, y_i) in terms of

$$(r - x_i), (s - y_i)$$

where (x_i, y_i) is a point near the root:

- **Taylor series expansion of functions**;

$$f(r, s) = 0 = f(x_i, y_i) + f_x(x_i, y_i)(r - x_i) + f_y(x_i, y_i)(s - y_i) + \dots$$
$$g(r, s) = 0 = g(x_i, y_i) + g_x(x_i, y_i)(r - x_i) + g_y(x_i, y_i)(s - y_i) + \dots$$

- Truncating both series gives

$$0 = f(x_i, y_i) + f_x(x_i, y_i)(r - x_i) + f_y(x_i, y_i)(s - y_i)$$
$$0 = g(x_i, y_i) + g_x(x_i, y_i)(r - x_i) + g_y(x_i, y_i)(s - y_i)$$

- which we can rewrite as

$$f_x(x_i, y_i)\Delta x_i + f_y(x_i, y_i)\Delta y_i = -f(x_i, y_i)$$
$$g_x(x_i, y_i)\Delta x_i + g_y(x_i, y_i)\Delta y_i = -g(x_i, y_i)$$

- where Δx_i and Δy_i are used as increments to x_i and y_i;

$$x_{i+1} = x_i + \Delta x_i$$

$$y_{i+1} = y_i + \Delta y_i$$

are improved estimates of the (x, y) values.

- We repeat this until both $f(x, y)$ and $g(x, y)$ are close to zero.

Example:

$f(x, y) = 4 - x^2 - y^2 = 0$
$g(x, y) = 1 - e^x - y = 0$

The partial derivatives are

$$f_x = -2x, f_y = -2y,$$

$$g_x = -e^x, g_y = -1$$

- Beginning with $x_0 = 1, y_0 = -1.7$, we solve

$$-2\Delta x_0 + 3.4\Delta y_0 = -0.1100$$
$$-2.7183\Delta x_0 - 1.0\Delta y_0 = 0.0183$$

- This gives

$\Delta x_0 = 0.0043,$
$\Delta y_0 = -0.0298$

- from which

$x_1 = 1.0043,$
$y_1 = -1.7298.$

- These agree with the true value within 2 in the fourth decimal place. Repeating the process once more:

  $x_2 = 1.004169,$
  $y_2 = -1.729637.$

  Then, f(1.004169,-1.729637)=-0.0000001,
  g(1.004169,-1.729637)=-0.00000001,

0.6.1 Solving a System by Iteration

- There is another way to attack a *system of nonlinear equations*.
- Consider this pair of equations:

equations;
$$e^x - y = 0,$$
$$xy - e^x = 0$$

rearrangement;
$$x = ln(y),$$
$$y = e^x/x$$

- We know how to solve *a single nonlinear equation by fixed-point* iterations
- We rearrange it to solve for the variable in a way that successive computations may reach a solution.
- To start, we guess at a value for y, say, $y = 2$. See Table 3.
- Final values are precisely the correct results.

Table 3: An example for solving a system by iteration

y-value	x-value
2	0.69315
2.88539	1.05966
2.72294	1.00171
2.71829	1.00000
2.71828	1.00000

Table 4: Another example for solving a system by iteration

y-value	x-value
-1.7291	1.0051
-1.72975	1.00398
-1.72961	1.00421
-1.72964	1.00416
-1.72963	1.00417

• **Example**: Another example for the pair of equations whose plot is Fig. 7.

equations;
$x^2 + y^2 = 4,$
$e^x + y = 1$

rearrangement; $y = -\sqrt{(4 - x^2)},$
$x = ln(1 - y)$

and begin with $x = 1.0$, the successive values for y and x are: (See Table 4)

• *We are converging to the solution in an oscillatory manner.*