3.1 The tolerance, tol, of the solution in the bisection method is given by $tol = \frac{1}{2}(b_n - a_n)$, where a_n and b_n are the endpoints of the interval after the nth iteration. The number of iterations n that are required for obtaining a solution with a tolerance that is equal to or smaller than a specified tolerance can be determined before the solution is calculated. Show that n is given by:

$$n \geq \frac{\log(b-a) - \log(tol)}{\log 2}$$

where a and b are the endpoints of the starting interval and tol is a user-specified tolerance.

Solution

$i = 1$ Starting interval $[a, b] = [a_1, b_1]$

$i = 2$ Second interval $[a_2, b_2]$: $b_2 - a_2 = \frac{1}{2}(b_1 - a_1)$

$i = 3$ Third interval $[a_3, b_3]$: $b_3 - a_3 = \frac{1}{2}(b_2 - a_2) = \frac{1}{4}(b_1 - a_1)$

$i = n$ interval $[a_n, b_n]$: $b_n - a_n = \left(\frac{1}{2}\right)^{n-1}(b_1 - a_1)$

Require: $\frac{1}{2}(b_n - a_n) \leq tol$

$$2tol \geq \left(\frac{1}{2}\right)^{n-1}(b_1 - a_1)$$

$$2^n \geq \frac{(b_1 - a_1)}{tol}$$

$$n \geq \frac{\log(b_1 - a_1) - \log(tol)}{\log 2}$$

3.2 Determine the root of $f(x) = x^3 - e^{-0.5x}$ by:

(a) Using the bisection method. Start with $a = 0$ and $b = 1$, and carry out the first four iterations.

(b) Using the secant method. Start with the two points, $x_1 = 0$, and $x_2 = 1$, and carry out the first four iterations.

(c) Using Newton's method. Start at $x_1 = 1$ and carry out the first four iterations.

Solution

Using 5 significant digits.

(a) $f(x) = x^3 - e^{-0.5x}$

$$i = 1, \quad a = 0, \quad b = 1, \quad f(0) = 0^3 - e^{-0.5(0)} = -1, \quad f(1) = 1^3 - e^{-0.5(1)} = 0.39347,$$

$$x_{NS1} = \frac{0+1}{2} = 0.5$$

$$f(0.5) = 0.5^3 - e^{-0.5(0.5)} = -0.6538$$

$$i = 2, \quad a = 0.5, \quad b = 1, \quad x_{NS2} = \frac{0.5+1}{2} = 0.75, \quad f(0.75) = 0.75^3 - e^{-0.5(0.75)} = -0.26541$$

$$i = 3, \quad a = 0.75, \quad b = 1, \quad x_{NS3} = \frac{0.75+1}{2} = 0.875, \quad f(0.875) = 0.875^3 - e^{-0.5(0.875)} = 0.024273$$

$$i = 4, \quad a = 0.75, \quad b = 0.875, \quad x_{NS4} = \frac{0.75+0.875}{2} = 0.8125,$$

$$f(0.8125) = 0.8125^3 - e^{-0.5(0.8125)} = -0.12977$$

(b) $x_{i+1} = x_i - \dfrac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$

$$x_1 = 0, \quad x_2 = 1, \quad f(x_1) = 0^3 - e^{-0.5(0)} = -1, \quad f(x_2) = 1^3 - e^{-0.5(1)} = 0.39347,$$

$$i = 2 \quad x_3 = 1 - \frac{0.39347(0-1)}{(-1) - 0.39347} = 0.71763, \quad f(x_3) = 0.71763^3 - e^{-0.5(0.71763)} = -0.32894$$

$$i = 3 \quad x_4 = 0.71763 - \frac{(-0.32894)(1 - 0.71763)}{0.39347 - (-0.32894)} = 0.84620, \quad f(x_4) = 0.84620^3 - e^{-0.5(0.84620)} = -0.049088$$

$$i = 4 \quad x_5 = 0.84620 - \frac{(-0.049088)(0.71763 - 0.84620)}{(-0.32894) - (-0.049088)} = 0.86875,$$

$$f(x_5) = 0.86875^3 - e^{-0.5(0.86875)} = 0.0079994$$

$$i = 5 \quad x_6 = 0.86875 - \frac{0.0079994(0.84620 - 0.86875)}{(-0.049088) - 0.0079994} = 0.86559$$

(c) $f(x) = x^3 - e^{-0.5x}$, $f'(x) = 3x^2 + 0.5e^{-0.5x}$, $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)} = x_i - \dfrac{x^3 - e^{-0.5x}}{3x^2 + 0.5e^{-0.5x}}$

$i = 1$, $x_1 = 1$, $x_2 = 1 - \dfrac{1^3 - e^{-0.5(1)}}{3(1^2) + 0.5e^{-0.5(1)}} = 0.88088$

$i = 2$, $x_3 = 0.88088 - \dfrac{0.88088^3 - e^{-0.5(0.88088)}}{3(0.88088^2) + 0.5e^{-0.5(0.88088)}} = 0.86618$

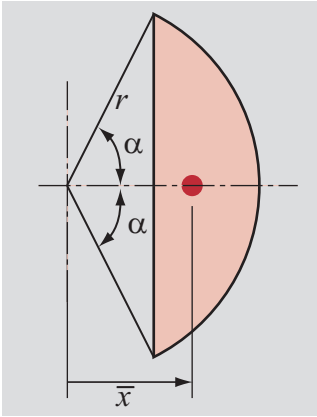
$i = 3$, $x_4 = 0.86618 - \dfrac{0.86618^3 - e^{-0.5(0.86618)}}{3(0.86618^2) + 0.5e^{-0.5(0.86618)}} = 0.86565$

$i = 4$, $x_5 = 0.86565 - \dfrac{0.86565^3 - e^{-0.5(0.86565)}}{3(0.86565^2) + 0.5e^{-0.5(0.86565)}} = 0.86565$

3.3 The location $\bar{x}$ of the centroid of a segment of a circle is given by:

$$\bar{x} = \frac{2r\sin^3\alpha}{3(\alpha - \sin\alpha\cos\alpha)}$$

Determine the angle α for which $\bar{x} = \frac{3r}{4}$.



First, derive the equation that must be solved and then determine the root using the following methods:

(*a*) Use the bisection method. Start with $a = 0.1$ and $b = 1.4$, and carry out the first four iterations.

(*b*) Use the secant method. Start with the two points $x_1 = 0.1$ and $x_2 = 1.4$, and carry out the first four iterations.

Solution

Using 5 significant digits.

(*a*) $$f(\alpha) = \frac{2\sin^3\alpha}{3(\alpha - \sin\alpha\cos\alpha)} - \frac{3}{4}$$

$$i = 1, \quad a = 0.1, \quad b = 1.4, \quad f(0.1) = \frac{2\sin^3 0.1}{3(0.1 - \sin 0.1\cos 0.1)} - \frac{3}{4} = 0.24700,$$

$$f(1.4) = \frac{2\sin^3 1.4}{3(1.4 - \sin 1.4\cos 1.4)} - \frac{3}{4} = -0.23237, \quad x_{NS1} = \frac{0.1 + 1.4}{2} = 0.75,$$

$$f(0.75) = \frac{2\sin^3 0.75}{3(0.75 - \sin 0.75\cos 0.75)} - \frac{3}{4} = 0.090352$$

$$i = 2, \quad a = 0.75, \quad b = 1.4, \quad x_{NS2} = \frac{0.75 + 1.4}{2} = 1.075,$$

$$f(1.075) = \frac{2\sin^3 1.075}{3(1.075 - \sin 1.075\cos 1.075)} - \frac{3}{4} = -0.058994$$

$$i = 3, \quad a = 0.75, \quad b = 1.075, \quad x_{NS3} = \frac{0.75 + 1.075}{2} = 0.9125,$$

$$f(0.9125) = \frac{2\sin^3 0.9125}{3(0.9125 - \sin 0.9125\cos 0.9125)} - \frac{3}{4} = 0.019976$$

$$i = 4, \quad a = 0.9125, \quad b = 1.075, \quad x_{NS4} = \frac{0.9125 + 1.075}{2} = 0.99375$$

(*b*) $\alpha_{i+1} = \alpha_i - \dfrac{f(\alpha_i)(\alpha_{i-1} - \alpha_i)}{f(\alpha_{i-1}) - f(\alpha_i)}$

$\alpha_1 = 0.1 \quad , \quad \alpha_2 = 1.4 \, , \quad f(0.1) = \dfrac{2\sin^3 0.1}{3(0.1 - \sin 0.1 \cos 0.1)} - \dfrac{3}{4} = 0.24700,$

$f(1.4) = \dfrac{2\sin^3 1.4}{3(1.4 - \sin 1.4 \cos 1.4)} - \dfrac{3}{4} = -0.23237 \, ,$

$i = 2 \quad \alpha_3 = 1.4 - \dfrac{(-0.23237)(0.1 - 1.4)}{0.24700 - -0.23237} = 0.76984 \, ,$

$f(0.76984) = \dfrac{2\sin^3 0.76984}{3(0.76984 - \sin 0.76984 \cos 0.76984)} - \dfrac{3}{4} = 0.82298$

$i = 3 \quad \alpha_4 = 0.76984 - \dfrac{0.82298(1.4 - 0.76984)}{(-0.23237) - 0.82298} = 0.93465 \, ,$

$f(0.93465) = \dfrac{2\sin^3 0.93465}{3(0.93465 - \sin 0.93465 \cos 0.93465)} - \dfrac{3}{4} = 0.0096639$

$i = 4 \quad \alpha_5 = 0.93465 - \dfrac{(0.0096639)(0.76984 - 0.93465)}{0.82298 - 0.0096639} = 0.95658 \, ,$

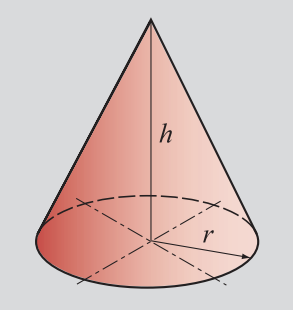
$f(0.95658) = \dfrac{2\sin^3 0.95658}{3(0.95658 - \sin 0.95658 \cos 0.95658)} - \dfrac{3}{4} = -6.9742 \times 10^{-4}$

$i = 5 \quad \alpha_6 = 0.95658 - \dfrac{(-6.9742 \times 10^{-4})(0.93465 - 0.95658)}{0.0096639 - -6.9742 \times 10^{-4}} = 0.95510$

3.4 The lateral surface area, S, of a cone is given by:

$$S = \pi r\sqrt{r^2 + h^2}$$

where r is the radius of the base and h is the height. Determine the radius of a cone that has a surface area of 1200 m^2 and a height of 20 m. Solve by using the fixed-point iteration method with $r = S/(\pi\sqrt{r^2 + h^2})$ as the iteration function. Start with $r = 17$ m and calculate the first four iterations.



Solution

Using 5 significant digits.

$$r_{i+1} = \frac{S}{\pi\sqrt{r_i^2 + h^2}}, \quad S = 1200\,\text{m}^2, \quad h = 20\ \text{m}$$

$$i = 1, \quad r_1 = 17, \quad r_2 = \frac{1200}{\pi\sqrt{17^2 + 20^2}} = 14.552$$

$$i = 2, \quad r_2 = 14.552, \quad r_3 = \frac{1200}{\pi\sqrt{14.552^2 + 20^2}} = 15.443$$

$$i = 3, \quad r_3 = 15.443, \quad r_4 = \frac{1200}{\pi\sqrt{15.443^2 + 20^2}} = 15.117$$

$$i = 4, \quad r_4 = 15.117, \quad r_5 = \frac{1200}{\pi\sqrt{15.117^2 + 20^2}} = 15.236$$

$$i = 5, \quad r_4 = 15.236, \quad r_5 = \frac{1200}{\pi\sqrt{15.236^2 + 20^2}} = 15.192$$

3.5 Determine the fifth root of 180 by finding the numerical solution of the equation $x^5 - 180 = 0$. Use Newton's method. Start at $x = 10$ and carry out the first seven iterations.

Solution

Using 5 significant digits.

$$f(x) = x^5 - 180, \quad f'(x) = 5x^4, \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^5 - 180}{5x_i^4}$$

$$i = 1, \quad x_1 = 180 \qquad x_2 = 180 - \frac{180^5 - 180}{5 \cdot 180^4} = 144$$

$$i = 2, \quad x_3 = 144 - \frac{144^5 - 144}{5 \cdot 144^4} = 115.2$$

$$i = 3, \quad x_4 = 115.2 - \frac{115.2^5 - 115.2}{5 \cdot 115.2^4} = 92.16$$

$$i = 4, \quad x_5 = 92.16 - \frac{92.16^5 - 92.16}{5 \cdot 92.16^4} = 73.728$$

$$i = 5, \quad x_6 = 73.728 - \frac{73.728^5 - 73.728}{5 \cdot 73.728^4} = 58.982$$

$$i = 6, \quad x_7 = 58.982 - \frac{58.982^5 - 58.982}{5 \cdot 58.982^4} = 47.186$$

$$i = 7, \quad x_7 = 47.186 - \frac{47.186^5 - 47.186}{5 \cdot 47.186^4} = 37.749$$

3.6 Determine the positive root of the polynomial $x^3 + 2.2x^2 - 8x - 16.8$.

(*a*) Plot the polynomial and choose a point near the root for the first estimate of the solution. Using Newton's method, determine the approximate solution in the first four iterations.

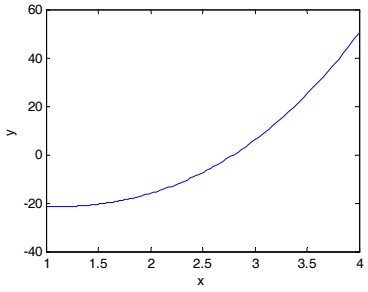
(*b*) From the plot in part (*a*), choose two points near the root to start the solution process with the secant method. Determine the approximate solution in the first four iterations.

Solution

(*a*)

Plot with MATLAB:

```
fplot('x^3+2.2*x^2-8*x-16.8',[1 4])
xlabel('x'); ylabel('y');
```



Solution with Newton's method start at $x_1 = 2$:

$$f(x) = x^3 + 2.2x^2 - 8x - 16.8, \quad f'(x) = 3x^2 + 4.4x - 8,$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^3 + 2.2x_i^2 - 8x_i - 16.8}{3x_i^2 + 4.4x_i - 8}$$

$$i = 1, \quad x_1 = 2 \qquad x_2 = 2 - \frac{2^3 + 2.2 \cdot 2^2 - (8 \cdot 2) - 16.8}{3 \cdot 2^2 + 4.4 \cdot 2 - 8} = 3.25$$

$$i = 2, \quad x_2 = 3.25 \qquad x_3 = 3.25 - \frac{3.25^3 + 2.2 \cdot 3.25^2 - (8 \cdot 3.25) - 16.8}{3 \cdot 3.25^2 + 4.4 \cdot 3.25 - 8} = 2.8613$$

$$i = 3, \quad x_3 = 2.8613 \qquad x_4 = 2.8613 - \frac{2.8613^3 + 2.2 \cdot 2.8613^2 - (8 \cdot 2.8613) - 16.8}{3 \cdot 2.8613^2 + 4.4 \cdot 2.8613 - 8} = 2.8014$$

$$i = 4, \quad x_4 = 2.8014 \qquad x_5 = 2.8014 - \frac{2.8014^3 + 2.2 \cdot 2.8014^2 - (8 \cdot 2.8014) - 16.8}{3 \cdot 2.8014^2 + 4.4 \cdot 2.8014 - 8} = 2.8000$$

(*b*) Solution with the secant method

$$f(x) = x^3 + 2.2x^2 - 8x - 16.8, \quad x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

$$x_1 = 2, \quad x_2 = 4, \quad f(x_1) = 2^3 + 2.2 \cdot 2^2 - (8 \cdot 2) - 16.8 = -16,$$

$$f(x_2) = 4^3 + 2.2 \cdot 4^2 - (8 \cdot 4) - 16.8 = 50.4,$$

$$i = 2 \qquad x_3 = 4 - \frac{50.4(2-4)}{(-16) - 50.4} = 2.4819, \quad f(x_3) = 2.4819^3 + 2.2 \cdot 2.4819^2 - (8 \cdot 2.4819) - 16.8 = -7.8155$$

$$i = 3 \qquad x_4 = 2.4819 - \frac{(-7.8155)(4 - 2.4819)}{50.4 - (-7.8155)} = 2.6857,$$

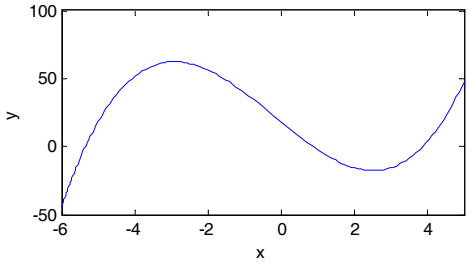
$$f(x_4) = 2.6857^3 + 2.2 \cdot 2.6857^2 - (8 \cdot 2.6857) - 16.8 = -3.0451$$

$$i = 4 \qquad x_5 = 2.6857 - \frac{(-3.0451)(2.4819 - 2.6857)}{-7.8155 - (-3.0451)} = 2.8158 \quad ,$$

$$f(x_5) = 2.8158^3 + 2.2 \cdot 2.8158^2 - (8 \cdot 2.8158) - 16.8 = 0.44252$$

$$i = 5 \qquad x_6 = 2.8158 - \frac{(0.44252)(2.6857 - 2.8158)}{-3.0451 - 0.44252} = 2.7993$$

3.7 Find all the roots of the equation $x^3 + 0.6x^2 - 22x + 18 = 0$ using the bisection method (four iterations for each root).



Solution

Plot with MATLAB:

```
fplot('x^3+0.6*x^2-22*x+18',[-6 5])
xlabel('x'); ylabel('y');
```

$f(x) = x^3 + 0.6x^2 - 22x + 18$

First root between $x = -6$ and $x = -4$

$i = 1, \quad a = -6, \quad b = -4,$

$f(-6) = (-6)^3 + 0.6(-6)^2 - 22(-6) + 18 = -44.4,$

$f(-4) = (-4)^3 + 0.6(-4)^2 - 22(-4) + 18 = 51.6, \quad x_{NS1} = \dfrac{-6 + (-4)}{2} = -5$

$f(-5) = (-5)^3 + 0.6(-5)^2 - 22(-5) + 18 = 18$

$i = 2, \quad a = -6, \quad b = -5, \quad x_{NS2} = \dfrac{-6 + (-5)}{2} = -5.5,$

$f(-5.5) = (-5.5)^3 + 0.6(-5.5)^2 - 22(-5.5) + 18 = -9.225$

$i = 3, \quad a = -5.5, \quad b = -5, \quad x_{NS3} = \dfrac{-5.5 + (-5)}{2} = -5.25,$

$f(-5.25) = (-5.25)^3 + 0.6(-5.25)^2 - 22(-5.25) + 18 = 5.3344$

$i = 4, \quad a = -5.5, \quad b = -5.25, \quad x_{NS4} = \dfrac{-5.5 + (-5.25)}{2} = -5.375,$

$f(-5.375) = (-5.375)^3 + 0.6(-5.375)^2 - 22(-5.375) + 18 = 1.7027$

$i = 5, \quad a = -5.4, \quad b = -5.375, \quad x_{NS5} = \dfrac{-5.4 + (-5.375)}{2} = -5.4375$

Second root between $x = 0$ and $x = 2$

$i = 1, \quad a = 0, \quad b = 2, \quad f(0) = (0)^3 + 0.6(0)^2 - 22(0) + 18 = 18,$

$f(2) = (2)^3 + 0.6(2)^2 - 22(2) + 18 = -15.6, \quad x_{NS1} = \dfrac{0 + 2}{2} = 1$

$f(1) = (1)^3 + 0.6(1)^2 - 22(1) + 18 = -2.4$

$i = 2, \quad a = 0, \quad b = 1, \quad x_{NS2} = \dfrac{0 + 1}{2} = 0.5,$

$f(0.5) = (0.5)^3 + 0.6(0.5)^2 - 22(0.5) + 18 = 7.275$

$i = 3, \quad a = 0.5, \quad b = 1, \quad x_{NS3} = \frac{0.5+1}{2} = 0.75,$

$f(0.75) = (0.75)^3 + 0.6(0.75)^2 - 22(0.75) + 18 = 2.2594$

$i = 4, \quad a = 0.75, \quad b = 1, \quad x_{NS4} = \frac{0.75+1}{2} = 0.875,$

$f(0.875) = (0.875)^3 + 0.6(0.875)^2 - 22(0.875) + 18 = -0.1207$

$i = 5, \quad a = 0.75, \quad b = 0.875, \quad x_{NS5} = \frac{0.75+0.875}{2} = 0.8125$

Third root between $x = 3$ and $x = 4$

$i = 1, \quad a = 3, \quad b = 4, \quad f(3) = (3)^3 + 0.6(3)^2 - 22(3) + 18 = -15.6,$

$f(4) = (4)^3 + 0.6(4)^2 - 22(4) + 18 = 3.6, \quad x_{NS1} = \frac{3+4}{2} = 3.5$

$f(3.5) = (3.5)^3 + 0.6(3.5)^2 - 22(3.5) + 18 = -8.775$

$i = 2, \quad a = 3.5, \quad b = 4, \quad x_{NS2} = \frac{3.5+4}{2} = 3.75,$

$f(3.75) = (3.75)^3 + 0.6(3.75)^2 - 22(3.75) + 18 = -3.3281$

$i = 3, \quad a = 3.75, \quad b = 4, \quad x_{NS3} = \frac{3.75+4}{2} = 3.875,$

$f(3.875) = (3.875)^3 + 0.6(3.875)^2 - 22(3.875) + 18 = -0.05508$

$i = 4, \quad a = 3.875, \quad b = 4, \quad x_{NS4} = \frac{3.875+4}{2} = 3.9375,$

$f(3.9375) = (3.9375)^3 + 0.6(3.9375)^2 - 22(3.9375) + 18 = 1.7240$

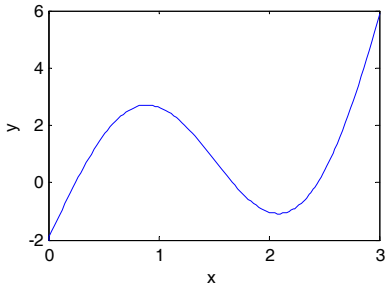
$i = 5, \quad a = 3.875, \quad b = 3.9375, \quad x_{NS5} = \frac{3.875+3.9375}{2} = 3.9063$

3.8 Find the first two positive roots of the equation $x^2 + 4\sin(2x) - 2 = 0$ using Newton's method (four iterations for each root).

Solution

Plot with MATLAB:

```
fplot('x^2+4*sin(2*x)-2',[0 3])
xlabel('x'); ylabel('y');
```



First solution with Newton's method start at $x_1 = 0$:

$$f(x) = x^2 + 4\sin(2x) - 2 \quad , \quad f'(x) = 2x + 8\cos(2x),$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 + 4\sin(2x_i) - 2}{2x_i + 8\cos(2x_i)}$$

$$i = 1, \quad x_1 = 0 \qquad x_2 = 0 - \frac{0^2 + 4\sin(2\cdot 0) - 2}{2\cdot 0 + 8\cos(2\cdot 0)} = 0.25$$

$$i = 2, \quad x_2 = 0.25 \qquad x_3 = 0.25 - \frac{0.25^2 + 4\sin(2\cdot 0.25) - 2}{2\cdot 0.25 + 8\cos(2\cdot 0.25)} = 0.25263$$

$$i = 3, \quad x_3 = 0.25263 \qquad x_4 = 0.25263 - \frac{0.25263^2 + 4\sin(2\cdot 0.25263) - 2}{2\cdot 0.25263 + 8\cos(2\cdot 0.25263)} = 0.25264$$

$$i = 4, \quad x_4 = 0.25264 \qquad x_5 = 0.25264 - \frac{0.25264^2 + 4\sin(2\cdot 0.25264) - 2}{2\cdot 0.25264 + 8\cos(2\cdot 0.25264)} = 0.25264$$

Second solution with Newton's method start at $x_1 = 1.9$:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 + 4\sin(2x_i) - 2}{2x_i + 8\cos(2x_i)}$$

$$i = 1, \quad x_1 = 1.9 \qquad x_2 = 1.9 - \frac{1.9^2 + 4\sin(2\cdot 1.9) - 2}{2\cdot 1.9 + 8\cos(2\cdot 1.9)} = 1.5687$$

$$i = 2, \quad x_2 = 1.5687 \qquad x_3 = 1.5687 - \frac{1.5687^2 + 4\sin(2\cdot 1.5687) - 2}{2\cdot 1.5687 + 8\cos(2\cdot 1.5687)} = 1.6669$$

$$i = 3, \quad x_3 = 1.6669 \qquad x_4 = 1.6669 - \frac{1.6669^2 + 4\sin(2\cdot 1.6669) - 2}{2\cdot 1.6669 + 8\cos(2\cdot 1.6669)} = 1.6701$$

$$i = 4, \quad x_4 = 1.6701 \qquad x_5 = 1.6701 - \frac{1.6701^2 + 4\sin(2\cdot 1.6701) - 2}{2\cdot 1.6701 + 8\cos(2\cdot 1.6701)} = 1.6701$$

3.9 Find four different iteration functions for solving the equation $3x^2 - x + e^{2x} - 2 = 0$ using the fixed-point iteration method.

Solution

$$x = \left(\frac{x - e^{2x} + 2}{3}\right)^{1/2}, \quad x = \frac{x - e^{2x} + 2}{3x}, \quad x = 3x^2 + e^{2x} - 2, \quad x = \frac{1}{2}\ln(-3x^2 + x + 2)$$

3.10 The equation $f(x) = x^2 - 5x^{1/3} + 1 = 0$ has a root between $x = 2$ and $x = 2.5$. To find the root by using the fixed-point iteration method, the equation has to be written in the form $x = g(x)$. Derive two possible forms for $g(x)$ — one by solving for x from the first term of the equation, and the next by solving for x from the second term of the equation.
(*a*) Determine which form should be used according to the condition in Eq. (3.30).
(*b*) Carry out the first five iterations using both forms of $g(x)$ to confirm your determination in part (*a*).

Solution

(*a*)
Solving for x from the first term of the equation:

$$x = (5x^{1/3} - 1)^{1/2}, \quad g(x) = (5x^{1/3} - 1)^{1/2}, \quad g'(x) = \frac{5}{3x^{2/3}(5x^{1/3} - 1)^{1/2}},$$

$g'(2) = 0.45608 \quad g'(2.5) = 0.37615$
Since for both $|g'| < 1$ the solution will converge.

Solving for x from the second term of the equation:

$$x = \left(\frac{x^2 + 1}{5}\right)^3, \quad g(x) = \left(\frac{x^2 + 1}{5}\right)^3, \quad g'(x) = 6\left(\frac{x^2 + 1}{5}\right)^2 x,$$

$g'(2) = 12 \quad g'(2.5) = 31.538$
Since for both $|g'| > 1$ the solution will not converge.
(*b*)
First five iterations using $x_{i+1} = (5x_i^{1/3} - 1)^{1/2}$ starting with $x = 2$:

$i = 1, \quad x_1 = 2, \quad x_2 = (5 \cdot 2^{1/3} - 1)^{1/2} = 2.3021$

$i = 2, \quad x_3 = (5 \cdot 2.3021^{1/3} - 1)^{1/2} = 2.3669$

$i = 3, \quad x_4 = (5 \cdot 2.3669^{1/3} - 1)^{1/2} = 2.3798$

$i = 4, \quad x_5 = (5 \cdot 2.3798^{1/3} - 1)^{1/2} = 2.3823$

First five iterations using $x_{i+1} = \left(\frac{x_i^2 + 1}{5}\right)^3$ starting with $x = 2.5$:

$i = 1, \quad x_1 = 2.5 \ , \quad x_2 = \left(\frac{2.5^2 + 1}{5}\right)^3 = 3.0486$

$i = 2, \quad x_3 = \left(\frac{3.0486^2 + 1}{5}\right)^3 = 8.7264$

$i = 3, \quad x_4 = \left(\frac{8.7264^2 + 1}{5}\right)^3 = 3.6737$

$i = 4, \quad x_5 = \left(\frac{3.6737^2 + 1}{5}\right)^3 = 24.369$

3.11 The equation $f(x) = x^3 + x^2 - 3x - 9 = 0$ has a root between $x = 2$ and $x = 2.5$. Find the root by using the fixed-point iteration method. Determine the appropriate form of $g(x)$ according to Eq. (3.30). Start the iterations with $x = 2.5$ and carry out the first five iterations.

Solution

$$x = (-x^2 + 3x + 9)^{1/3} \quad g(x) = (-x^2 + 3x + 9)^{1/3} \quad g'(x) = \frac{-(2x - 3)}{3(-x^2 + 3x + 9)^{2/3}}$$

$$g'(2) = \frac{-(2 \cdot 2 - 3)}{3(-2^2 + 3 \cdot 2 + 9)^{2/3}} = -0.067393 \quad g'(2.5) = \frac{-(2 \cdot 2.5 - 3)}{3(-2.5^2 + 3 \cdot 2.5 + 9)^{2/3}} = -0.14128$$

$$x_{i+1} = (-x_i^2 + 3x_i + 9)^{1/3}$$

$$i = 1, \quad x_1 = 2.5, \quad x_2 = (-2.5^2 + 3 \cdot 2.5 + 9)^{1/3} = 2.1722$$

$$i = 2, \quad x_3 = (-2.1722^2 + 3 \cdot 2.1722 + 9)^{1/3} = 2.2103$$

$$i = 3, \quad x_4 = (-2.2103^2 + 3 \cdot 2.2103 + 9)^{1/3} = 2.2067$$

$$i = 4, \quad x_5 = (-2.2067^2 + 3 \cdot 2.2067 + 9)^{1/3} = 2.2070$$

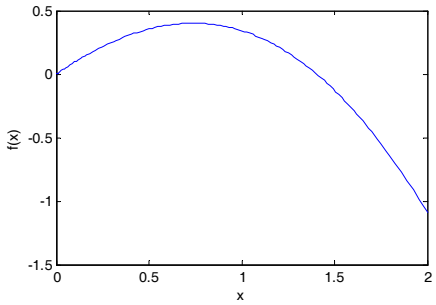
$$i = 5, \quad x_5 = (-2.2070^2 + 3 \cdot 2.2070 + 9)^{1/3} = 2.2070$$

3.12 Determine the positive root of the equation $\sin x - 0.5x^2 = 0$ by using the fixed-point iteration

Solution

Plot with MATLAB:

```
fplot('x^3+0.6*x^2-22*x+18',[-6 5])
xlabel('x'); ylabel('f(x)');
```



$$x = (2\sin x)^{1/2} \qquad g(x) = (2\sin x)^{1/2}$$

$$g'(x) = \frac{\cos x}{(2\sin x)^{1/2}}$$

$$g'(1.5) = \frac{\cos 1.5}{(2\sin 1.5)^{1/2}} = 0.05008$$

$$x_{i+1} = (2\sin x_i)^{1/2}$$

$i = 1,\quad x_1 = 1.5\ ,\quad x_{i+1} = (2\sin 1.5)^{1/2} = 1.4124$

$i = 2,\quad x_{i+1} = (2\sin 1.4124)^{1/2} = 1.4053$

$i = 3,\quad x_{i+1} = (2\sin 1.4053)^{1/2} = 1.4045$

$i = 4,\quad x_{i+1} = (2\sin 1.4045)^{1/2} = 1.4044$

3.13 Solve the following system of nonlinear equations:

$$4x^2 - y^3 + 28 = 0$$

$$3x^3 + 4y^2 - 145 = 0$$

(*a*) Use Newton's method. Start at $x = 1$, $y = 1$, and carry out the first five iterations.

(b) Use the fixed-point iteration method. Use the iteration functions $y = (4x^2 + 28)^{1/3}$ and $x = \left[\frac{(145 - 4y^2)}{3}\right]^{1/3}$. Start at $x = 1$, $y = 1$, and carry out the first five iterations.

Solution

(*a*) Using Newton's method.

$f_1 = 4x^2 - y^3 + 28 = 0$

$f_2 = 3x^3 + 4y^2 - 145 = 0$

The Jacobian (Eq. (3.47)) is:

$$\frac{\partial f_1}{\partial x} = 8x, \quad \frac{\partial f_1}{\partial y} = -3y^2, \quad \frac{\partial f_2}{\partial x} = 9x^2, \quad \frac{\partial f_2}{\partial y} = 8y$$

$$J(f_1, f_2) = \det\begin{bmatrix} 8x & -3y^2 \\ 9x^2 & 8y \end{bmatrix} = 64xy + 27x^2y^2$$

Substituting in Eqs. (3.45) and (3.46):

$$\Delta x = \frac{-(4x^2 - y^3 + 28)(8y) + (3x^3 + 4y^2 - 145)(-3y^2)}{64xy + 27x^2y^2}, \quad \Delta y = \frac{-(3x^3 + 4y^2 - 145)(8x) + (4x^2 - y^3 + 28)9x^2}{64xy + 27x^2y^2}$$

The solution in each iteration is given by Eqs. (3.84): $x_{i+1} = x_i + \Delta x_i$, $y_{i+1} = y_i + \Delta y_i$

The first iteration is calculated by hand:

$$i = 1, \quad x_1 = 1, \quad y_1 = 1, \quad J(f_1, f_2) = 64 \cdot 1 \cdot 1 + 27 \cdot 1^2 \cdot 1^2 = 91$$

$$\Delta x = \frac{-(4 \cdot 1^2 - 1^3 + 28)(8 \cdot 1) + (3 \cdot 1^3 + 4 \cdot 1^2 - 145)(-3 \cdot 1^2)}{91} = 1.8242$$

$$\Delta y = \frac{-(3 \cdot 1^3 + 4 \cdot 1^2 - 145)(8 \cdot 1) + (4 \cdot 1^2 - 1^3 + 28)(9 \cdot 1^2)}{91} = 15.198$$

$$x_2 = 1 + 1.8242 = 2.8242, \quad y_2 = 1 + 15.198 = 16.198$$

The calculations of the first 5 iterations are calculated with the following MATLAB program (script file).

```
%Solution of Chapter 3 HW 13a
clear all
F1 = @ (x,y)4*x^2-y^3+28;
F2 = @ (x,y)3*x^3+4*y^2-145;
```

2

```
F1x = @ (x) 8*x;
F1y = @ (y) -3*y^2;
F2x = @ (x) 9*x^2;
F2y = @ (y) 8*y;
Jacob = @ (x,y) 64*x*y+27*x^2*y^2;
xi = 1; yi = 1;
fprintf('x = %-7.4f y = %-7.4f\n',xi,yi)
for i = 1:5
    Jac = Jacob(xi,yi);
    Delx = (-F1(xi,yi)*F2y(yi) + F2(xi,yi)*F1y(yi))/Jac;
    Dely = (-F2(xi,yi)*F1x(xi) + F1(xi,yi)*F2x(xi))/Jac;
    xi = xi + Delx;
    yi = yi + Dely;
    fprintf('i =%2.0f x = %-7.4f y = %-7.4f\n',i,xi,yi)
end
```

When the program is executed, the following (solution) is displayed in the Command Window:

```
x = 1.0000  y = 1.0000
i = 1 x = 2.8242  y = 16.1978
i = 2 x = -0.9143 y = 10.7673
i = 3 x = -3.5495 y = 7.3238
i = 4 x = -1.8115 y = 5.0630
i = 5 x = 3.0737  y = 2.9895
```

If the program is executed for a few more iterations the solution is converged to $x = 3, \quad y = 4$.

(*b*) Using the fixed-point iteration method.
The iteration function are:

Use the iteration functions $y_{i+1} = (4x_i^2 + 28)^{1/3}$ and $x_{i+1} = \left[\frac{(145 - 4y_i^2)}{3}\right]^{1/}$

$i = 1, \quad x_1 = 1, \quad y_1 = 1, \quad x_2 = \left[\frac{(145 - (4 \cdot 1^2))}{3}\right]^{1/3} = 3.6088, \quad y_2 = (4 \cdot 1^2 + 28)^{1/3} = 3.1748$

$i = 2, \quad x_3 = \left[\frac{(145 - (4 \cdot 3.1748^2))}{3}\right]^{1/3} = 3.2678, \quad y_3 = (4 \cdot 3.6088^2 + 28)^{1/3} = 4.3106$

$$i = 3, \quad x_4 = \left[\frac{(145 - (4 \cdot 4.3106^2))}{3}\right]^{1/3} = 2.8667, \quad y_4 = (4 \cdot 3.2678^2 + 28)^{1/3} = 4.1353$$

$$i = 4, \quad x_5 = \left[\frac{(145 - (4 \cdot 4.1353^2))}{3}\right]^{1/3} = 2.9446, \quad y_5 = (4 \cdot 2.8667^2 + 28)^{1/3} = 3.9337$$

$$i = 5, \quad x_6 = \left[\frac{(145 - (4 \cdot 3.9337^2))}{3}\right]^{1/3} = 3.0258, \quad y_6 = (4 \cdot 2.9446^2 + 28)^{1/3} = 3.9724$$

3.14 Solve the following system of nonlinear equations:

$$x^2 + 2x + 2y^2 - 26 = 0$$
$$2x^3 - y^2 + 4y - 19 = 0$$

(*a*) Use Newton's method. Start at $x = 1$, $y = 1$, and carry out the first five iterations.
(*b*) Use the fixed-point iteration method. Start at $x = 1$, $y = 1$, and carry out the first five iterations.

Solution

(*a*) Using Newton's method.

$f_1 = x^2 + 2x + 2y^2 - 26 = 0$

$f_2 = 2x^3 - y^2 + 4y - 19 = 0$

The Jacobian (Eq. (3.47)) is:

$$\frac{\partial f_1}{\partial x} = 2x + 2, \quad \frac{\partial f_1}{\partial y} = 4y, \quad \frac{\partial f_2}{\partial x} = 6x^2, \quad \frac{\partial f_2}{\partial y} = -2y + 4$$

$$J(f_1, f_2) = \det\begin{bmatrix} 2x+2 & 4y \\ 6x^2 & -2y+4 \end{bmatrix} = (2x+2)(-2y+4) - 24x^2y = -2xy + 8x - 4y + 8 - 24x^2y$$

Substituting in Eqs. (3.45) and (3.46):

$$\Delta x = \frac{-(x^2 + 2x + 2y^2 - 26)(-2y + 4) + (2x^3 - y^2 + 4y - 19)(4y)}{-2xy + 8x - 4y + 8 - 24x^2y},$$

$$\Delta y = \frac{-(2x^3 - y^2 + 4y - 19)(2x + 2) + (x^2 + 2x + 2y^2 - 26)6x^2}{-2xy + 8x - 4y + 8 - 24x^2y}$$

The solution in each iteration is given by Eqs. (3.84): $x_{i+1} = x_i + \Delta x_i$, $y_{i+1} = y_i + \Delta y_i$

The first iteration is calculated by hand:

$$i = 1, \quad x_1 = 1, \quad y_1 = 1, \quad J(f_1, f_2) = -2 \cdot 1 \cdot 1 + 8 \cdot 1 - (4 \cdot 1) + 8 - (24 \cdot 1^2 \cdot 1) = -16$$

$$\Delta x = \frac{-(1^2 + 2 \cdot 1 + 2 \cdot 1^2 - 19)(-2 \cdot 1 + 4) + (2 \cdot 1^3 - 1^2 + 4 \cdot 1 - 26)(4 \cdot 1)}{-16} = 0.875$$

$$\Delta y = \frac{-(2 \cdot 1^3 - 1^2 + 4 \cdot 1 - 19)(2 \cdot 1 + 2) + (1^2 + 2 \cdot 1 + 2 \cdot 1^2 - 26)(6 \cdot 1^2)}{-16} = 4.375$$

$$x_2 = 1 + 0.875 = 1.875, \quad y_2 = 1 + 4.375 = 5.375$$

The calculations of the first 5 iterations are calculated with the following MATLAB program (script file).

```
clear all
F1 = @ (x,y) x^2+2*x+2*y^2-26;
F2 = @ (x,y) 2*x^3-y^2+4*y-19;
```

2

```
F1x = @ (x) 2*x+2;
F1y = @ (y) 4*y;
F2x = @ (x) 6*x^2;
F2y = @ (y) -2*y+4;
Jacob = @ (x,y) (2*x+2)*(-2*y+4)-4*y*6*x^2;
xi = 1; yi = 1;
fprintf('x = %-7.4f y = %-7.4f\n',xi,yi)
for i = 1:5
    Jac = Jacob(xi,yi);
    Delx = (-F1(xi,yi)*F2y(yi) + F2(xi,yi)*F1y(yi))/Jac;
    Dely = (-F2(xi,yi)*F1x(xi) + F1(xi,yi)*F2x(xi))/Jac;
    xi = xi + Delx;
    yi = yi + Dely;
    fprintf('i =%2.0f x = %-7.4f y = %-7.4f\n',i,xi,yi)
end
```

When the program is executed, the following (solution) is displayed in the Command Window:

```
x = 1.0000  y = 1.0000
i = 1 x = 1.8750  y = 5.3750
i = 2 x = 1.9164  y = 3.5478
i = 3 x = 1.9963  y = 3.0443
i = 4 x = 2.0000  y = 3.0003
i = 5 x = 2.0000  y = 3.0000
```

3.15 In the program of Example 3-1 the iterations are executed in the for-end loop. In the loop, the anonymous function *F* is used twice (once in the command `FxNS = F(xNS)` and once in the command `if F(a)*FxNS < 0`). Rewrite the program such that the anonymous function *F* is used inside the loop only once. Execute the new program and show that the output is the same as in the example.

Solution

```
clear all
F = @ (x) 8-4.5*(x-sin(x));
a = 2; b = 3; imax = 20; tol = 0.001;
Fa=F(a); Fb=F(b);
if Fa*Fb > 0
    disp('Error: The function has the same sign at points a and b.')
else
    disp('iteration     a          b      (xNS) Solution   f(xNS)    Tolerance')
    for i = 1:imax
        xNS = (a + b)/2;
        toli=(b-a)/2;
        FxNS=F(xNS);
        fprintf('%3i    %11.6f %11.6f %11.6f  %11.6f %11.6f\n',i, a, b, xNS,
FxNS, toli)
        if FxNS == 0
            fprintf('An exact solution x =%11.6f was found',xNS)
            break
        end
        if toli < tol
            break
        end
        if i == imax
        fprintf('Solution was not obtained in %i iterations',imax)
        break
        end
        if Fa*FxNS < 0
```

```
            b = xNS;
        else
        a = xNS;
        Fa = FxNS;
        end
    end
end
```

When the program is executed the display in the Command Window is:

```
iteration     a           b       (xNS) Solution   f(xNS)     Tolerance
  1       2.000000    3.000000    2.500000    -0.556875    0.500000
  2       2.000000    2.500000    2.250000     1.376329    0.250000
  3       2.250000    2.500000    2.375000     0.434083    0.125000
  4       2.375000    2.500000    2.437500    -0.055709    0.062500
  5       2.375000    2.437500    2.406250     0.190661    0.031250
  6       2.406250    2.437500    2.421875     0.067838    0.015625
  7       2.421875    2.437500    2.429688     0.006154    0.007813
  8       2.429688    2.437500    2.433594    -0.024755    0.003906
  9       2.429688    2.433594    2.431641    -0.009295    0.001953
 10       2.429688    2.431641    2.430664    -0.001569    0.000977
```

3.16 Write a MATLAB user-defined function that solves for a root of a nonlinear equation $f(x) = 0$ using the bisection method. Name the function `Xs = BisectionRoot(Fun,a,b,TolMax)`. The output argument `Xs` is the solution. The input argument `Fun` is a name for the function that calculates $f(x)$ for a given x (it is a dummy name for the function that is imported into `BisectionRoot`), `a` and `b` are two points that bracket the root, and `TolMax` is the maximum tolerance.
The program should include the following features:

- Check if points `a` and `b` are on opposite sides of the solution. If not, the program should stop and display an error message.
- The number of iterations should be determined (using the equation in Problem 3.1) before the iterations are carried out.

Use `BisectionRoot` to solve the equation in Example 3-1. Use the value of 0.00001 for `TolMax`.

Solution

```
function Xs = BisectionRoot(Fun,a,b,TolMax)
% BisectionRoot finds the root of Fun = 0 using the bisection method.
% Input variables:
% Fun  Name for the function (entered as a handle) that calculates Fun for a
given x.
% a, b   Two points that bracket the root.
% TolMax The maximum tolerance.
% Output variable:
% Xs     Solution
Fa = Fun(a);
Fb = Fun(b);
%  Check if points a and b are on opposite sides of the solution.
if Fa*Fb > 0
    Xs = ('Error: The function has the same sign at points a and b.');
else
%  Calculate the number of iterations.
    n = ceil((log10(b-a)-log10(TolMax))/log10(2));
    for i = 1:n
        Xs = (a + b)/2;
        FXs=Fun(Xs);
```

```
        if FXs == 0
            break
        end
        if Fa*FXs < 0
            b = Xs;
        else
            a = Xs;
            Fa = FXs;
        end
    end
end
```

To solve the equation in Example 3-1, an anonymous function that calculates the value of the function $f(x) = 8 - 4.5(x - \sin x)$, is defined.

The user-defined function `BisectionRoot` is then used to solve the equation:

```
>> F= @ (x) 8-4.5*(x-sin(x));
>> x=BisectionRoot(F,2,3,0.00001)
x =
   2.430473327636719
```

3.17 Determining the square root of a number p, $\sqrt{p}$, is the same as finding a solution to the equation $f(x) = x^2 - p = 0$. Write a MATLAB user-defined function that determines the square root of a positive number by solving the equation using Newton's method. Name the function `[Xs] = SquareRoot(p)`. The output argument `Xs` is the answer, and the input argument `p` is the number whose square root is determined. The program should include the following features:

- It should check if the number is positive. If not, the program should stop and display an error message.
- The starting value of x for the iterations should be $x = p$.
- The iterations should stop when the estimated relative error (Eq. (3.9)) is smaller than 0.00001.
- The number of iterations should be limited to 20. If a solution is not obtained in 20 iterations, the program should stop and display an error message.

Use the function `SquareRoot` to determine the square root of (*a*) 729, (*b*) 1500, and (*c*) -72.

Solution

```
function Xs = SquareRoot(p)
% SquareRoot finds the root of a number.
% Input variables:
% p  The number whose square root is determined.
% Output variable:
% Xs  The square root of p.

Xest = p; imax = 20;
if p < 0
    disp('ERROR Input argument cannot be negative.')
    Xs = 'Error';
else
    for i = 1:imax
    Xi = Xest - (Xest^2-p)/(2*Xest);
        if abs((Xi - Xest)/Xest) < 0.00001
        Xs = Xi;
        break
        end
    Xest = Xi;
```

```
    end
    if i == imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs = ('No answer');
    end
end
```

The function `SquareRoot` is next used in the Command Window to determine the square root of (*a*) 729, (*b*) 1500, and (*c*) -72.

```
>> SquareRoot(729)
ans =
  27.00000000000000
>> SquareRoot(1500)
ans =
  38.72983346232766
>> SquareRoot(-72)
ERROR Input argument cannot be negative.
ans =
Error
```

3.18 Determining the cubic root of a number p, $\sqrt[3]{p}$, is the same as finding a solution to the equation $f(x) = x^3 - p = 0$. Write a MATLAB user-defined function that determines the cubic root of a number by solving the equation using Newton's method. Name the function `Xs = CubeRoot(p)`. The output argument `Xs` is the answer, and the input argument `p` is the number whose cubic root is determined.
The program should include the following features:

- The starting value of x for the iterations should be $x = p$.
- The iterations should stop when the estimated relative error (Eq. (3.9)) is smaller than 0.00001.
- The number of iterations should be limited to 20. If a solution is not obtained in 20 iterations, the program should stop and display an error message.

Use the function `CubeRoot` to determine the cubic root of (*a*) 13824, (*b*) 1408, and (*c*) -600.

Solution

```
function Xs = cubeRoot(p)
% CubeRoot finds the cubic root of a number.
% Input variables:
% p  The number whose cubic root is determined.
% Output variable:
% Xs  The cubic root of p.

Xest = p; imax = 50;
    for i = 1:imax
    Xi = Xest - (Xest^3-p)/(3*Xest^2);
        if abs((Xi - Xest)/Xest) < 0.00001
        Xs = Xi;
        break
        end
    Xest = Xi;
    end
    if i == imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs = ('No answer');
    end
```

```
end
```

```
>> Xs = cubeRoot(13824)
Xs =
  24.000000000000007
>> Xs = cubeRoot(1408)
Xs =
  11.208157322622606
>> Xs = cubeRoot(-600)
Xs =
  -8.434326653017493
>>
```

3.19 A new method for solving a nonlinear equation $f(x) = 0$ is proposed. The method is similar to the bisection method. The solution starts by finding an interval $[a, b]$ that brackets the solution. The first estimate of the solution is the midpoint between $x = a$ and $x = b$. Then the interval $[a, b]$ is divided into four equal sections. The section that contains the root is taken as the new interval for the next iteration.

Write a MATLAB user-defined function that solves a nonlinear equation with the proposed new method. Name the function `Xs = QuadSecRoot(Fun,a,b,TolMax)`, where the output argument `Xs` is the solution. The input argument `Fun` is a name for the function that calculates $f(x)$ for a given x (it is a dummy name for the function that is imported into `QuadSecRoot`), `a` and `b` are two points that bracket the root, and `TolMax` is the maximum tolerance.

Use the user-defined `QuadSecRoot` function to solve the equations in Problems 3.2 and 3.3. For the initial values of a and b, take the values that are listed in part (a) of the problems, and for `TolMax`, use 0.00001.

Solution

```
function Xs = QuadSecRoot(Fun,a,b,TolMax)
% QuadSecRoot finds the root of Fun = 0.
% Input variables:
% Fun  Name for the function (entered as a handle) file that calculates Fun for
a given x.
% a, b   Two points that bracket the root.
% TolMax The maximum tolerance.
% Output variable:
% Xs     Solution
imax = 20;
Fa = Fun(a);
Fb = Fun(b);
%  Check if points a and b are on opposite sides of the solution.
if Fa*Fb > 0
    Xs = ('Error: The function has the same sign at points a and b.');
else
    for i = 1:imax
    Xs = (a + b)/2;
    toli=(b-a)/2;
```

```
    Fxs=Fun(Xs);
        if toli < TolMax
            break
        end
        if i == imax
        fprintf('Solution was not obtained in %i iterations',imax)
        break
        end
%Divide the interval into three sections.
    d=(b-a);
    aa=a+d/4;
    am=a+d/2;
    bb=a+3*d/4;
    Faa=Fun(aa);
    Fam=Fun(am);
    Fbb=Fun(bb);
        if Fa*Faa < 0
            b = aa;
            Fb = Faa;
        elseif Faa*Fam < 0
            a = aa; b=am;
            Fa = Faa; Fb = Fam;
        elseif Fam*Fbb < 0
            a = am; b=bb;
            Fa = Fam; Fb = Fbb;
        else
            a=bb;
            Fa = Fbb;
        end
    end
end
```

To solve the equations in Problems 3.2 and 3.3, anonymous functions that calculate the value of the functions $f(x) = x^3 - e^{-0.5x}$ and $f(\alpha) = \dfrac{2\sin^3\alpha}{3(\alpha - \sin\alpha\cos\alpha)} - \dfrac{3}{4}$ are created.

The function `QuadSecRoot` is then used in the Command Window:

```
>> F3_2= @ (x) x^3-exp(-0.5*x);
>> Xs = QuadSecRoot(F3_2,0,1,0.00001)
Xs =
   0.865653991699219
>> F3_3= @ (x) 2*sin(x)^3/(3*(x-sin(x)*cos(x)))-3/4;
>> Xs = QuadSecRoot(F3_3,0.1,1.4,0.00001)
Xs =
   0.955118560791016
```

3.20 Write a MATLAB user-defined function that solves a nonlinear equation $f(x) = 0$ with the regula falsi method. Name the function `Xs = RegulaRoot(Fun,a,b,ErrMax)`, where the output argument `Xs` is the solution. The input argument `Fun` is a name for the function that calculates $f(x)$ for a given x (it is a dummy name for the function that is imported into `RegulaRoot`), `a` and `b` are two points that bracket the root, and `ErrMax` the maximum error according to Eq. (3.9).

The program should include the following features:

- Check if points `a` and `b` are on opposite sides of the solution. If not, the program should stop and display an error message.
- The number of iterations should be limited to 100 (to avoid an infinite loop). If a solution with the required accuracy is not obtained in 100 iterations, the program should stop and display an error message.

Use the function `RegulaRoot` to solve the equation in Problem 3.3 (use $a = 0.1$, $b = 1.4$). For `ErrMax` use 0.00001.

Solution

```
function Xs = RegulaRoot(Fun,a,b,ErrMax)
% RegulaRoot finds the root of Fun = 0 using the regula falsi method.
% Input variables:
% Fun  Name for the function (imported as a handle) that calculates Fun for a
given x.
% a, b   Two points that bracket the root.
% TolMax The maximum error.
% Output variable:
% Xs     Solution
imax = 100;
Fa = Fun(a);
Fb = Fun(b);
%  Checks if points a and b are on opposite sides of the solution.
if Fa*Fb > 0
    Xs = ('Error: The function has the same sign at points a and b.');
else
    for i = 1:imax
        if i >1
```

```
            XsIminus1 = Xs;
        end
        Xs = (a*Fb-b*Fa)/(Fb-Fa);
        FXs=Fun(Xs);
        if FXs == 0
            break
        end
% Ckecks if the estimated relative error is smaller than ErrMax.
        if i >1 & abs((Xs-XsIminus1)/XsIminus1) <= ErrMax
            break
        end
        if i == imax
            fprintf('Solution was not obtained in %i iterations',imax)
        break
        end
        if Fa*FXs < 0
            b = Xs;
            Fb = FXs;
        else
            a = Xs;
            Fa = FXs;
        end
    end
end
```

Solution of Problem 3.3 (Command Window):

```
>> F3_3= @ (x) 2*sin(x)^3/(3*(x-sin(x)*cos(x)))-3/4;
>> Xs = RegulaRoot(F3_3,0.1,1.4,0.00001)
Xs =
   0.955113485152213
```

3.21 A new method for solving a nonlinear equation $f(x) = 0$ is proposed. The method is a combination of the bisection and the regula falsi methods. The solution starts by defining an interval $[a, b]$ that brackets the solution. Then estimated numerical solutions are determined once with the bisection method and once with the regula falsi method. (The first iteration uses the bisection method.) Write a MATLAB user-defined function that solves a nonlinear equation $f(x) = 0$ with this new method. Name the function `Xs = BiRegRoot(Fun,a,b,ErrMax)`, where the output argument `Xs` is the solution. The input argument `Fun` is a name for the function that calculates $f(x)$ for a given x (it is a dummy name for the function that is imported into `BiRegRoot`), `a` and `b` are two points that bracket the root, and `ErrMax` is the maximum error according to Eq. (3.9).

The program should include the following features:

- Check if points `a` and `b` are on opposite sides of the solution. If not, the program should stop and display an error message.
- The number of iterations should be limited to 100 (to avoid an infinite loop). If a solution with the required accuracy is not obtained in 100 iterations, the program should stop and display an error message.

Use the function `RegulaRoot` to solve the equation in Problem 3.3 (use $a = 0.1$, $b = 1.4$). For `ErrMax` use 0.00001.

Solution

```
function Xs = BiRegRoot(Fun,a,b,ErrMax)
% BiRegRootT finds the root of Fun = 0 using a combination of the bisection
% and regula falsi methods
% Input variables:
% Fun  Name for the function (imported as a handle) that calculates Fun for a
given x.
% a, b   Two points that bracket the root.
% TolMax The maximum tolerance.
% Output variable:
% Xs     Solution
imax = 100;
Fa = Fun(a);
Fb = Fun(b);
%  Check if points a and b are on opposite sides of the solution.
```

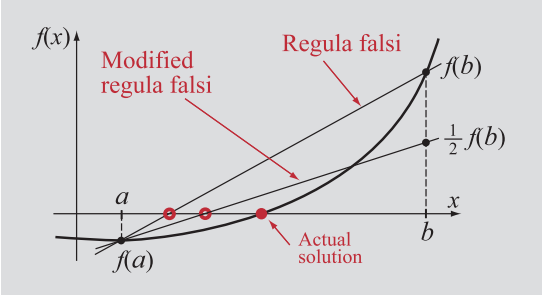
```
if Fa*Fb > 0
    Xs = ('Error: The function has the same sign at points a and b.');
else

    for i = 1:imax
        if i >1
            XsIminus1 = Xs;
        end
        if rem(i,2)==0
        Xs = (a*Fb-b*Fa)/(Fb-Fa);
        else
        Xs = (a + b)/2;
        end
        FXs=Fun(Xs);
        if FXs == 0
            break
        end
        % Ckecks if the estimated relative error is smaller than ErrMax.
        if i >1 & abs((Xs-XsIminus1)/XsIminus1) <= ErrMax
            break
        end
        if i == imax
            fprintf('Solution was not obtained in %i iterations',imax)
        break
        end
        if Fa*FXs < 0
            b = Xs;
        else
            a = Xs;
            Fa = FXs;
        end
```

```
    end
end
```

Solution of Problem 3.3 (Command Window):

```
>> F3_3= @ (x) 2*sin(x)^3/(3*(x-sin(x)*cos(x)))-3/4;
>> Xs = BiRegRoot(F3_3,0.1,1.4,0.00001)
Xs =
   0.9543280489152643
```

3.22 Frequently, in the regula falsi method (see Section 3.4), one of the endpoints of the interval stays the same in all the iterations, while the other endpoint advances toward the root. In the modified regula falsi method, when this situation occurs, the straight line that connects the endpoints of the interval is replaced with a line that has a smaller slope. As shown in the figure, this is done by dividing by 2 the value of the function at the endpoint that stays the same. Consequently, the line intersects the x-axis closer to the root. Write a MATLAB user-defined function that solves a nonlinear equation $f(x) = 0$ with the modified regula falsi method. The program should check if the endpoints stay the same; if an endpoint is unchanged in three iterations, the value of the function at that point should be divided by 2, when Eq. (3.11) is used. If the same end point is unchanged for three more iterations, the value of the function at that point should be divided again by 2 (the true value now is divided by 4), and so on. Name the function `Xs = RegulaRootMod(Fun,a,b,ErrMax)`. The function arguments and features are the same as in Problem 3.20.



Use the function `RegulaRootMod` to solve the equation in Problem 3.3 (use $a = 0.1$, $b = 1.4$). For `ErrMax` use 0.0001.

Solution

```
function Xs = RegulaRootMod(Fun,a,b,ErrMax)
% RegulaRootMod finds the root of Fun = 0 using the modified regula falsi
method.
% Input variables:
% Fun  Name for the function (imported as handle) that calculates Fun for a
given x.
% a, b   Two points that bracket the root.
% TolMax The maximum error.
% Output variable:
% Xs     Solution
imax = 100;
Fa = Fun(a); ia = 0; Fac = Fa; ai = a;
Fb = Fun(b); ib = 0; Fbc = Fb; bi = b;
%  Checks if points a and b are on opposite sides of the solution.
```

```
if Fa*Fb > 0
    Xs = ('Error: The function has the same sign at points a and b.');
else
    for i = 1:imax
        if i >1
            XsIminus1 = Xs;
        end
        if ai == a
            ia = ia+1;
            na = ia/3;
            if rem(ia,3)== 0
                Fac = Fa/(2*na);
            end
        else
            ai=a;
            ia=1;
        end
        if bi == b
            ib=ib+1;
            nb = ib/3;
            if rem(ib,3)== 0
                Fbc = Fb/(2*nb);
            end
        else
            bi = b;
            ib = 1;
        end
        Xs = (a*Fbc-b*Fac)/(Fbc-Fac);
        FXs=Fun(Xs);
        if FXs == 0
            break
```

```
        end
% Ckecks if the estimated relative error is smaller than ErrMax.
        if i >1 & abs((Xs-XsIminus1)/XsIminus1) <= ErrMax
            break
        end
        if i == imax
            fprintf('Solution was not obtained in %i iterations',imax)
        break
        end
        if Fa*FXs < 0
            b = Xs;
            Fb = FXs; Fbc=Fb;
        else
            a = Xs;
            Fa = FXs; Fac=Fa;
        end
    end
end
```

Solution of Problem 3.3 (Command Window):

```
>> F3_3= @ (x) 2*sin(x)^3/(3*(x-sin(x)*cos(x)))-3/4;
>> Xs = RegulaRootMod(F3_3,0.1,1.4,0.0001)
Xs =
   0.955114080820920
```

3.23 Modify the function `NewtonRoot` that is listed in Fig. 3-11, such that the output will have three arguments. Name the function `[Xs,FXs,iact] = NewtonRootMod(Fun,FunDer,Xest,Err,imax)`. The first output argument is the solution, the second is the value of the function at the solution, and the third is the actual number of iterations that are performed to obtain the solution. Use the function `NewtonRootMod` to solve the equation that is solved in Example 3-2.

Solution

```
function [Xs,FXs,iact] = NewtonRootMod(Fun,FunDer,Xest,Err,imax)
% NewtonRoot finds the root of Fun = 0 near the point Xest using Newton's
method.
% Input variables:
% Fun    Name for the function (imported as a handle) that calculates Fun for a
given x.
% FunDir Name for the function (imported as a handle) that calculates the
derivative of
%        Fun for a given x.
% Xest   Initial estimate of the solution.
% Err    Maximum error.
% imax   Maximum number of iterations.
% Output variable:
% Xs     Solution.
% FXs    The value of the function at the solution.
% iact   The actual number of iterations that are performed.

for i = 1:imax
    Xi = Xest - Fun(Xest)/FunDer(Xest);
    if abs((Xi - Xest)/Xest) < Err
        Xs = Xi;
        FXs = Fun(Xest);
        iact = i;
        break
    end
    Xest = Xi;
```

```
end
if i == imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs = ('No answer');
end
```

To solve the equation in Example 3-2, the following user-defined functions that calculate $f(x) = 8 - 4.5(x - \sin x)$ and $f'(x) = -4.5(1 - \cos x)$ are created.

```
function y = FunExample2(x)
y = 8 - 4.5*(x - sin(x));

function y = FunDerSample2(x)
y = -4.5 + 4.5*cos(x);
```

The function `NewtonRootMod` is then used in the Command Window:

```
>> [A,B,C] = NewtonRootMod(@FunExample2,@FunDerExample2,2,0.0001,10)
A =
   2.430465741723630
B =
   -3.991472770081828e-007
C =
     4
```

3.24 Write a user-defined MATLAB function that solves for all the real roots in a specified domain of a nonlinear function $f(x) = 0$ using the bisection method. Name the function `R=BisecAll-Roots(fun,a,b,TolMax)`. The output argument `R` is a vector whose elements are the values of the roots. The input argument `Fun` is a name for a function that calculates $f(x)$ for a given x. (It is a dummy name for the function that is imported into `BisecAllRoots`.) The arguments `a` and `b` define the domain, and `TolMax` is the maximum tolerance that is used by the bisection method when the value of each root is calculated. Use the following algorithm:

1. Divide the domain $[a, b]$ into 10 equal subintervals of length h such that $h = (b - a)/10$.
2. Check for a sign change of $f(x)$ at the endpoints of each subinterval.
3. If a sign change is identified in a subinterval, use the bisection method for determining the root in that subinterval.
4. Divide the domain $[a, b]$ into 100 equal subintervals of length h such that $h = (b - a)/100$.
5. Repeat step 2. If a sign change is identified in a subinterval, check if it contains a root that was already obtained. If not, use the bisection method for determining the root in that subinterval.
6. If no new roots have been identified, stop the program.
7. If one or more new roots have been identified, repeat steps 4–6, wherein each repetition the number of subintervals is multiplied by 10.

Use the function `BisecAllRoots`, with `TolMax` value of 0.0001, to find all the roots of the equation $x^4 - 5.5x^3 - 7.2x^2 + 43x + 36 = 0$.

Solution

```
function R = BisecAllRoots(Fun,a,b,TolMax)
% BisecAllRoot finds all the roots of Fun = 0 in the interval [a,b]
% using the bisection method.
% Input variables:
% Fun  Name for the function (imported as a handle) that calculates Fun for a
given x.
% a, b   Two points that define the interval [a,b].
% TolMax The maximum tolerance for each root.
% Output variable:
% R      Vector with the roots.
n = 10;   % Number of subintervals.
c = 1;
```

```
is = 1;
j = 0;
while c==1
    c = 0;
    h = (b - a)/n;
    j=j+1;
    if j==5    % j is used to prevent an indefinite while loop.
        break
    end
    for i=1:n
        ai=a+h*(i-1);
        bi=ai+h;
        Fai = Fun(ai);
        Fbi = Fun(bi);
% Cgecking for a sign change in a subinterval.
        if Fai*Fbi < 0   %If a sign change is detected, calculate the root.
            xs=BisectionRoot(Fun,ai,bi,TolMax);
% Assign the root to vector R.
            if n==10 & is==1
                R(is)=xs;
                is=is+1;
            else
                nr=length(R);
                e=0;
                for ka=1:nr   % Checking if the root is already known.
                    if abs(R(ka)-xs)<TolMax
                        e=1;
                    end
                end
                if e==0   %  If the root is not known, add it to R.
                    R(is)=xs;
```

```
                    is=is+1;
                    c=1;
                end
            end

        end
    end
 n = 10*n;    % Increase the number of subitervals by a factor of 10.
end
```

To find the roots of the equation $x^4 - 5.5x^3 - 7.2x^2 + 43x + 36 = 0$ the following user-defined function is created.

```
function y = FunHW3_24(x)
y = x^4-5.5*x^3-7.2*x^2+43*x+36;
```

The function `RegulaRootMod` is then used in the Command Window:

```
>> R = BisecAllRoots(@FunHW3_24,-10,10,0.0001)
R =
-2.392517089843750  -0.805236816406250   3.873596191406250   4.824157714843750
```

3.25 Examine the differences between the True Relative Error, Eq. (3.8), and the Estimated Relative Error, Eq. (3.9), by numerically solving the equation $f(x) = 0.5e^{(2+x)} - 40 = 0$. The exact solution of the equation is $x = \ln(80) - 2$. Write a MATLAB program in a script file that solves the equation by using Newton's method. Start the iterations at $x = 4$, and execute 11 iterations. In each iteration, calculate the True Relative Error (TRE) and the Estimated Relative Error (ERE). Display the results in a four-column table (create a 2-dimensional array), with the number of iterations in the first column, the estimated numerical solution in the second, and TRE and ERE in the third and fourth columns, respectively.

Solution

Script file:

```
clear all
F=@ (x) 0.5*exp(2+x)-40;
FD=@ (x) 0.5*exp(2+x);
xT=log(80)-2;
fprintf('Exact solution    %19.11e\n',xT)
x(1)=4;
disp('Iteration    Num Sol           TRE               ERE')
for i = 1:11
    x(i+1) = x(i) - F(x(i))/FD(x(i));
     TRE(i)=(xT - x(i+1))/xT;
     ERE(i)=(x(i+1)-x(i))/x(i);
     fprintf('%3i    %19.11e %19.11e %19.11e\n',i, x(i+1), TRE(i), ERE(i))
end
```

When the script file is executed, the display in the Command Window is:

```
Exact solution     2.38202663467e+000
Iteration    Num Sol           TRE               ERE
  1     3.19830017413e+000 -3.42680273838e-001 -2.00424956467e-001
  2     2.64037614165e+000 -1.08457858202e-001 -1.74443923992e-001
  3     2.41270139352e+000 -1.28775885191e-002 -8.62281492941e-002
  4     2.38249233123e+000 -1.95504345465e-004 -1.25208458726e-002
  5     2.38202674309e+000 -4.55157846570e-008 -1.95420624099e-004
  6     2.38202663467e+000 -2.42363357488e-015 -4.55157801617e-008
```

```
  7     2.38202663467e+000 -3.72866703827e-016 -2.05076687105e-015
  8     2.38202663467e+000 -1.86433351914e-016 -1.86433351914e-016
  9     2.38202663467e+000 -3.72866703827e-016  1.86433351914e-016
 10     2.38202663467e+000 -1.86433351914e-016 -1.86433351914e-016
 11     2.38202663467e+000 -3.72866703827e-016  1.86433351914e-016
```

3.26 When calculating the payment of a mortgage, the relationship between the loan amount, *Loan*, the monthly payment, *MPay*, the duration of the loan in months *Months*, and the annual interest rate, *Rate*, is given by the equation (annuity equation):

$$MPay = \frac{Loan \cdot Rate}{12\left(1 - \frac{1}{\left(1 + \frac{Rate}{12}\right)^{Months}}\right)}$$

Determine the duration of a $200,000 loan at an interest rate of 5.5% (enter 0.055 in the equation) if the monthly payment is $1460.60.

(*a*) Use the user-defined function `BisectionRoot` from Problem 3.16. Use 0.0001 for `TolMax`.

(*b*) Use MATLAB's built-in function `fzero`.

Solution

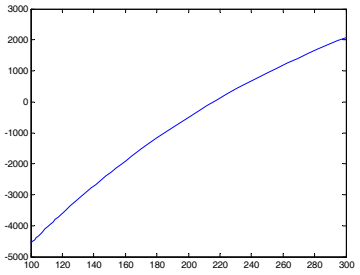
(*a*) The function is: $f(x) = 1460.6 \cdot 12\left(1 - \frac{1}{\left(1 + \frac{0.055}{12}\right)^{x}}\right) - (200000 \cdot 0.055)$, which can be written as:

$$f(x) = 6527.2 - \frac{17527.2}{\left(1 + \frac{0.055}{12}\right)^{x}}.$$

A plot of the function (shown on the right) for the domain $[100, 300]$ is obtained with MATLAB by typing:

```
>> F=@(x) 6527.2-17527.2/(1+0.055/12)^x;
>> fplot(F,[100 300])
```

The root appears to be in the neighborhood of $x = 200$.



(*a*) A solution with the user-defined function `BisectionRoot` from Problem 3.16, using 200 for `a` and 300 for `b`, and 0.0001 for `TolMax`. (Command Window):

```
>> Xs = BisectionRoot(F,200,300,0.0001)
Xs =
  216.0083
>>
```

The duration is 216 months.

(*b*) A solution with MATLAB's built-in function `fzero` is (Command Window):

```
>> duration=fzero(F,200)
duration =
  216.0083
```

The duration is 216 months.