Solutions

Chapter 1 Getting Started

1.1 Algorithms

1. Use the statistics algorithm from the text to compute the mean, $\bar{x}$, and the standard deviation, s, of the data set: $-5, -3, 2, -2, 1$.

The inputs are

$$x_1 = -5, \quad x_2 = -3, \quad x_3 = 2, \quad x_4 = -2, \quad x_5 = 1, \quad \text{and} \quad n = 5.$$

Working sequentially through the steps of the algorithm, we find

STEP 1:	$xsum = 0;$	$x2sum = 0$
STEP 2:	$i = 1:$	$xsum = 0 + (-5) = -5;$ $x2sum = 0 + (-5)^2 = 25$
	$i = 2:$	$xsum = -5 + (-3) = -8;$ $x2sum = 25 + (-3)^2 = 34$
	$i = 3:$	$xsum = -8 + 2 = -6;$ $x2sum = 34 + 2^2 = 38$
	$i = 4:$	$xsum = -6 + (-2) = -8;$ $x2sum = 38 + (-2)^2 = 42$
	$i = 5:$	$xsum = -8 + 1 = -7;$ $x2sum = 42 + 1^2 = 43$
STEP 3:	$xbar = -7/5 = -1.4$	
STEP 4:	$s = \sqrt{(5(43) - (-7)^2)/(5 \cdot 4)} = 2.880972$	
OUTPUT:	$xbar = -1.4$ and $s = 2.880972$	

Thus, for the data set consisting of the five numbers -5, -3, 2, -2, and 1, the mean is $\bar{x} = -1.4$ and the standard deviation is $s = 2.880972$.

2. With $n = 4$, use the trapezoidal rule algorithm from the text to approximate the value of the definite integral

$$\int_0^1 \frac{1}{1+x^2}\,dx.$$

Matching this specific problem to the general pattern $\int_a^b f(x)\,dx$, we see that $a = 0$, $b = 1$ and $f(x) = \frac{1}{1+x^2}$. With $n = 4$, the trapezoidal rule algorithm yields

STEP 1: $h = (1-0)/4 = 1/4; \quad sum = 0$
STEP 2: $i = 1$: $sum = 0 + 1/(1+(1/4)^2) = 16/17$
$i = 2$: $sum = 16/17 + 1/(1+(1/2)^2) = 148/85$
$i = 3$: $sum = 148/85 + 1/(1+(3/4)^2) = 1012/425$
STEP 3: $sum = 2(1012/425) = 2024/425$
STEP 4: $sum = 2024/425 + 1/1 + 1/2 = 5323/850$
OUTPUT: $(1/8)(5323/850) = 5323/6800 = 0.782794117$

Therefore,

$$\int_0^1 \frac{1}{1+x^2}\,dx \approx 0.782794117.$$

The exact value of the integral is $\pi/4$, so the absolute error in the trapezoidal rule approximation is $|\pi/4 - 0.782794117| \approx 2.604 \times 10^{-3}$.

3. Use the square root algorithm from the text to approximate $\sqrt{5}$. Take $x_0 = 5$, $\epsilon = 5 \times 10^{-4}$ and $Nmax = 10$.

Let $a = 5$, $x_0 = 5$, $\epsilon = 5 \times 10^{-4}$, and $Nmax = 10$. The first iteration of the square root algorithm yields

STEP 1: $iter = 1$
STEP 2: $x_1 = (5 + 5/5)/2 = 3$
STEP 3: $|x_1 - x_0| = 2 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 3$

In the second iteration, we find

STEP 1: $iter = 2$
STEP 2: $x_1 = (3 + 5/3)/2 = 7/3$
STEP 3: $|x_1 - x_0| = 2/3 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 7/3$

In the third iteration, we find

STEP 1: $iter = 3$
STEP 2: $x_1 = (7/3 + 15/7)/2 = 47/21$
STEP 3: $|x_1 - x_0| = 2/21 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 47/21$

The fourth iteration yields

STEP 1: $iter = 4$
STEP 2: $x_1 = (47/21 + 105/47)/2 = 2207/987$
STEP 3: $|x_1 - x_0| = 2/987 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 2207/987$

and the fifth iteration produces

STEP 1: $iter = 5$
STEP 2: $x_1 = (2207/987 + 4935/2207)/2 = 4870847/2178309$
STEP 3: $|x_1 - x_0| = 2/2178309 \approx 9.18 \times 10^{-7} < 5 \times 10^{-4}$, so
OUTPUT $x_1 = 4870847/2178309 \approx 2.236067977$

Thus, $\sqrt{5} \approx 2.236067977$.

4. A different scheme for approximating the square root of a positive real number a is based on the recursive formula

$$x_{n+1} = \frac{x_n^3 + 3x_n a}{3x_n^2 + a}.$$

(a) Construct an algorithm for approximating the square root of a given positive real number a using this formula.

(b) Test your algorithm using $a = 2$ and $x_0 = 2$. Allow a maximum of 10 iterations and use a convergence tolerance of $\epsilon = 5 \times 10^{-5}$. Compare the performance of this algorithm with the one presented in the text.

(a)

GIVEN: nonnegative real number *a*
starting approximation x_0
convergence parameter ϵ
maximum number of iterations $Nmax$

STEP 1: for *iter* from 1 to *Nmax*
STEP 2: compute $x_1 = (x_0^3 + 3ax_0)/(3x_0^2 + a)$
STEP 3: if $|x_1 - x_0| < \epsilon$, OUTPUT x_1
STEP 4: copy the value of x_1 to x_0
end
OUTPUT: "maximum number of iterations exceeded"

(b) Let $a = 2$, $x_0 = 2$, $\epsilon = 5 \times 10^{-5}$, and $Nmax = 10$. The first iteration of the algorithm from part (a) yields

STEP 1: $iter = 1$
STEP 2: $x_1 = (8 + 12)/(12 + 2) = 10/7$
STEP 3: $|x_1 - x_0| = 4/7 > 5 \times 10^{-5}$, so
STEP 4: $x_0 = 10/7$

In the second iteration, we find

STEP 1: $iter = 2$
STEP 2: $x_1 = (1000/343 + 60/7)/(300/49 + 2) = 3940/2786$
STEP 3: $|x_1 - x_0| = 40/2786 > 5 \times 10^{-5}$, so
STEP 4: $x_0 = 3940/2786$

The third iteration then yields

STEP 1: $iter = 3$
STEP 2: $x_1 = 1.414213562$
STEP 3: $|x_1 - x_0| \approx 3.64 \times 10^{-7} < 5 \times 10^{-5}$, so
OUTPUT $x_1 = 1.414213562$

Thus, $\sqrt{2} \approx 1.414213562$, which is correct to the digits shown. Observe that with the same number of iterations (three), the current algorithm produced a more accurate approximation than the algorithm from the text.

5. Let A be an $n \times m$ matrix and B be an $m \times p$ matrix. The $n \times p$ matrix $C = AB$ has elements defined by

$$c_{ik} = \sum_{j=1}^{m} a_{ij} b_{jk}$$

for each $i = 1, 2, 3, \ldots, n$ and each $k = 1, 2, 3, \ldots, p$. Construct an algorithm to compute the product of two matrices.

GIVEN: the integers *n*, *m* and *p*
the elements in the matrix *A*, a_{ij}
the elements in the matrix *B*, b_{jk}

STEP 1: for *i* from 1 to *n*
STEP 2: for *k* from 1 to *p*
STEP 3: initialize *sum* to 0
STEP 4: for *j* from 1 to *m*
add $a_{ij}b_{jk}$ to *sum*
end
STEP 5: set $c_{ik} = sum$
end
end
OUTPUT: the elements in the matrix *C*, c_{ik}

6. Consider the computation of the following sum,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j,$$

where the a_i and b_j are real numbers.

(a) How many multiplications and how many additions are required to compute the sum? Each answer should be a function of n.

(b) Modify the summation to an equivalent form which reduces the number of operations needed. How many multiplications and how many additions are required to compute the sum in its revised form?

(a) As written, forming the product of a_i and b_j for each possible combination of i and j requires n^2 multiplications. Summing all n^2 terms requires $n^2 - 1$ additions.

(b) Observe that

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j = \left(\sum_{i=1}^{n} a_i\right)\left(\sum_{j=1}^{n} b_j\right).$$

Using the expression on the right-hand side, the original sum can now be computed using only $2n - 2$ additions and a single multiplication.

7. Let a be a non-zero real number. For any x_0 satisfying $0 < x_0 < 2/a$, the recursive sequence defined by

$$x_{n+1} = x_n(2 - ax_n)$$

converges to $1/a$.

(a) Construct an algorithm for approximating the reciprocal of a given non-zero real number a using this formula.

(b) Test your algorithm using $a = 37$ and $x_0 = 0.01$. Allow a maximum of 10 iterations and use a convergence tolerance of $\epsilon = 5 \times 10^{-4}$.

(a)

GIVEN: non-zero real number *a*
starting approximation x_0
convergence parameter ϵ
maximum number of iterations *Nmax*

STEP 1: for *iter* from 1 to *Nmax*
STEP 2: compute $x_1 = x_0(2 - ax_0)$
STEP 3: if $|x_1 - x_0| < \epsilon$, OUTPUT x_1
STEP 4: copy the value of x_1 to x_0
end
OUTPUT: "maximum number of iterations has been exceeded"

(b) Let $a = 37$, $x_0 = 0.01$, $\epsilon = 5 \times 10^{-4}$, and $Nmax = 10$. The first iteration of the algorithm from part (a) yields

STEP 1: $iter = 1$
STEP 2: $x_1 = 0.01(2 - 0.01 \cdot 37) = 0.0163$
STEP 3: $|x_1 - x_0| = 0.0063 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 0.0163$

In the second iteration, we find

STEP 1: $iter = 2$
STEP 2: $x_1 = 0.0163(2 - 0.0163 \cdot 37) = 0.02276947$
STEP 3: $|x_1 - x_0| = 0.00646947 > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 0.02276947$

The third iteration then yields

STEP 1: $iter = 3$
STEP 2: $x_1 = 0.026356335$
STEP 3: $|x_1 - x_0| \approx 3.59 \times 10^{-3} > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 0.026356335$

In the fourth iteration, we calculate

STEP 1: $iter = 4$
STEP 2: $x_1 = 0.027010383$
STEP 3: $|x_1 - x_0| \approx 6.54 \times 10^{-4} > 5 \times 10^{-4}$, so
STEP 4: $x_0 = 0.027010383$

With the fifth iteration, we find

STEP 1: $iter = 5$
STEP 2: $x_1 = 0.027027016$
STEP 3: $|x_1 - x_0| \approx 1.66 \times 10^{-5} < 5 \times 10^{-4}$, so
OUTPUT $x_1 = 0.027027016$

Thus, $1/37 \approx 0.027027016$, which is in error by roughly 1.10×10^{-7}.

8. Given two positive integers a and b, the greatest common divisor of a and b is the largest integer which divides both a and b; *i.e.*, the largest integer n for which both a/n and b/n are integers.

(a) Construct an algorithm to compute the greatest common divisor of two positive integers.

(b) How many divisions does your algorithm require?

(a)

GIVEN: positive integers *a* and *b*

STEP 1: initialize $c = \min(a, b)$ and $gcd = 1$
STEP 2: for *n* from 2 to *c*
STEP 3: if a/n and b/n are integers, set $gcd = n$
OUTPUT: gcd

(b) The above algorithm requires $2(c - 1)$ divisions, where $c = \min(a, b)$.

9. The inner product, or dot product, of two n-vectors $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 + x_3 y_3 + \cdots + x_n y_n.$$

(a) Construct an algorithm to compute the inner product of two n-vectors.

(b) Apply your algorithm to calculate the inner product of the vectors

$$\mathbf{x} = \begin{bmatrix} -3 & 4 & 1 & 2 \end{bmatrix}^T \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 & -3 & 2 & 5 \end{bmatrix}^T$$

(a) GIVEN: n-vectors $\mathbf{x}$ and $\mathbf{y}$
vector dimension n

STEP 1: initialize $sum = 0$
STEP 2: for i from 1 to n
add $x_i \cdot y_i$ to sum
OUTPUT: sum

(b) The inputs are the vectors

$$\mathbf{x} = [\ -3 \quad 4 \quad 1 \quad 2\ ] \qquad \text{and} \qquad \mathbf{y} = [\ 1 \quad -3 \quad 2 \quad 5\ ]$$

and $n = 4$. Working sequentially through the steps of the algorithm, we find

STEP 1: $sum = 0$
STEP 2: $i = 1$: $sum = 0 + (-5)(1) = -3$
$i = 2$: $sum = -3 + (4)(-3) = -15$
$i = 3$: $sum = -15 + (1)(2) = -13$
$i = 4$: $sum = -13 + (2)(5) = -3$
OUTPUT: $sum = -3$

Thus, $\mathbf{x} \cdot \mathbf{y} = -3$.

10. The linear correlation coefficient for n ordered pairs (x_i, y_i) is given by the formula

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}.$$

(a) Construct an algorithm to compute the linear correlation coefficient for a given set of ordered pairs.

(b) Apply your algorithm to compute the linear correlation coefficient for the following set of ordered pairs:

x_i	3	7	9	2	7	0	3
y_i	-5	10	15	-8	11	-10	-4

(a) GIVEN: ordered pairs (x_i, y_i)
number of ordered pairs n

STEP 1: initialize $xsum$, $ysum$, $x2sum$, $y2sum$ and $xysum$ to 0
STEP 2: for i from 1 to n
add x_i to $xsum$
add y_i to $ysum$
add x_i^2 to $x2sum$
add y_i^2 to $y2sum$
add $x_i \cdot y_i$ to $xysum$

end

STEP 3: calculate $r = \dfrac{n \cdot xysum - (xsum)(ysum)}{\sqrt{n \cdot x2sum - (xsum)^2} \cdot \sqrt{n \cdot y2sum - (ysum)^2)}}$

OUTPUT: r

(b) Working sequentially through the steps of the algorithm, we find

STEP 1: $xsum = ysum = x2sum = y2sum = xysum = 0$

STEP 2:

$i = 1$: $xsum = 0 + 3 = 3$; $ysum = 0 + (-5) = -5$
$x2sum = 0 + 3^2 = 9$; $y2sum = 0 + (-5)^2 = 25$
$xysum = 0 + (3)(-5) = -15$

$i = 2$: $xsum = 3 + 7 = 10$; $ysum = -5 + 10 = 5$
$x2sum = 9 + 7^2 = 58$; $y2sum = 25 + 10^2 = 125$
$xysum = -15 + (7)(10) = 55$

$i = 3$: $xsum = 10 + 9 = 19$; $ysum = 5 + 15 = 20$
$x2sum = 58 + 9^2 = 139$; $y2sum = 125 + 15^2 = 350$
$xysum = 55 + 9(15) = 190$

$i = 4$: $xsum = 19 + 2 = 21$; $ysum = 20 + (-8) = 12$
$x2sum = 139 + 2^2 = 143$; $y2sum = 350 + (-8)^2 = 414$
$xysum = 190 + (2)(-8) = 174$

$i = 5$: $xsum = 21 + 7 = 28$; $ysum = 12 + 11 = 23$
$x2sum = 143 + 7^2 = 192$; $y2sum = 414 + 11^2 = 535$
$xysum = 174 + (7)(11) = 251$

$i = 6$: $xsum = 28 + 0 = 28$; $ysum = 23 + (-10) = 13$
$x2sum = 192 + 0^{=}192$; $y2sum = 535 + (-10)^2 = 635$
$xysum = 251 + (0)(-10) = 251$

$i = 7$: $xsum = 28 + 3 = 31$; $ysum = 13 + (-4) = 9$
$x2sum = 192 + 3^2 = 201$; $y2sum = 635 + (-4)^2 = 651$
$xysum = 251 + (3)(-4) = 239$

STEP 3: $r = \dfrac{7(239) - (31)(9)}{\sqrt{7(201) - (31)^2} \cdot \sqrt{7(651) - (9)^2)}} = 0.987$

OUTPUT: $r = 0.987$

Thus, the linear correlation coefficient of the seven given data pairs is 0.987.

11. The midpoint rule approximates the value of a definite integral using the formula

$$\int_a^b f(x)dx \approx 2h \sum_{j=1}^{n} f(x_j),$$

where $h = (b - a)/2n$ and $x_j = a + (2j - 1)h$.

(a) Construct an algorithm to approximate the value of a definite integral using the midpoint rule.

(b) Apply your algorithm to approximate the value of $\int_1^2 dx/x$. Take $n = 4$. Compare the approximation obtained from the midpoint rule with the approximation obtained from the trapezoidal rule.

(a)

GIVEN: the limits of integration *a* and *b*
the integrand *f*
the number of subintervals *n*

STEP 1: compute $h = (b-a)/(2n)$; initialize $sum = 0$
STEP 2: for *j* from 1 to *n*
add $f(a + (2j-1)h)$ to sum
OUTPUT: $2h \cdot sum$

(b) Matching this specific problem to the general pattern $\int_a^b f(x)\,dx$, we see that $a = 1$, $b = 2$ and $f(x) = \frac{1}{x}$. With $n = 4$, the midpoint rule algorithm yields

STEP 1: $h = (2-1)/(2 \cdot 4) = 1/8$; $sum = 0$
STEP 2: $j = 1$: $sum = 0 + 1/(1 + 1/8) = 8/9$
$j = 2$: $sum = 8/9 + 1/(1 + 3/8) = 160/99$
$j = 3$: $sum = 160/99 + 1/(1 + 5/8) = 2872/1287$
$j = 4$: $sum = 2872/1287 + 1/(1 + 7/8) = 17792/6435$
OUTPUT: $2(1/8)(17792/6435) = 4448/6435 = 0.691219891$

Therefore,

$$\int_1^2 \frac{1}{x}\,dx \approx 0.691219891.$$

The exact value of the integral is $\ln 2$, so the absolute error in the midpoint rule approximation is $|\ln 2 - 0.691219891| \approx 1.927 \times 10^{-3}$. This error is roughly half the error in the trapezoidal rule approximation obtained in the text.

12. Consider the following algorithm for the trapezoidal rule:

GIVEN: the limits of integration a and b
the integrand f
the number of subintervals n
STEP 1: compute $h = (b-a)/n$; and initialize sum to 0
STEP 2: for i from 1 to $n-1$
add $2f(a + ih)$ to sum
STEP 3: add $f(a)$ and $f(b)$ to sum
OUTPUT: $(h/2)sum$

Compare the number of arithmetic operations required by this algorithm to the number of operations required by the algorithm presented in the text.

Aside from the operations needed to evaluate f, the trapezoidal rule algorithm presented in the text uses $2n + 1$ additions/subtractions (one in STEP 1, $2(n-1)$

in STEP 2 and two in STEP 4) and $n+3$ multiplications/divisions (one in STEP 1, $n-1$ in STEP 2, one in STEP 3 and two in the OUTPUT). The modified algorithm presented in this exercise uses the exact same additions/subtractions, but requires a total of $2n+1$ multiplications/divisions (one in STEP 1, $2(n-1)$ in STEP 2 and 2 in the OUTPUT).

13. Rewrite the algorithm for the trapezoidal rule which was presented in the text to reduce both the number of additions and the number of multiplications/divisions by one.

The key is to observe that

$$\frac{h}{2}\left[f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right]=h\left[\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n-1}f(x_i)\right].$$

This suggests modifying the trapezoidal rule algorithm to

GIVEN: the limits of integration *a* and *b*
the integrand *f*
the number of subintervals *n*

STEP 1: compute $h=(b-a)/n$; initialize $sum=(f(a)+f(b))/2$
STEP 2: for *i* from 1 to *n-1*
add $f(a+ih)$ to sum
OUTPUT: $h\cdot sum$

This algorithm uses $2n$ additions/subtractions (two in STEP 1 and $2(n-1)$ in STEP 2) and $n+2$ multiplications/divisions (two in STEP 1, $n-1$ in STEP 2 and one in the OUTPUT).

14. Let $P(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ be an n-th degree polynomial with all real coefficients, and let x_0 be a given real number.

(a) Treating integer powers as repeated multiplication, how many multiplications and how many additions are required to evaluate $P(x)$ at $x=x_0$?

(b) Devise an algorithm for computing the value of an n-th degree polynomial which reduces the required number of arithmetic operations? How many multiplications and how many additions are required by your algorithm?

(a) For each j, evaluation of the term $a_jx_0^j$ requires j multiplications ($j-1$ to evaluate the power of x_0 and one more to include the coefficient). Thus, to evaluate the entire polynomial requires

$$\sum_{j=1}^{n}j=\frac{n(n+1)}{2}$$

multiplications. Because the polynomial consists of $n+1$ terms, a total of n additions are needed for evaluation.

(b) To devise a more efficient scheme for evaluating an nth-degree polynomial, we rewrite the polynomial in nested form:

$$\begin{aligned} P(x) &= a_0 + x(a_1 + a_2x + a_3x^2 + \cdots a_nx^{n-1}) \\ &= a_0 + x(a_1 + x(a_2 + a_3x + \cdots + a_nx^{n-2})) \\ &= \cdots \\ &= a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots x(a_{n-1} + a_nx)\cdots))). \end{aligned}$$

Based on this form, we can evaluate the polynomial with the following algorithm:

GIVEN: the polynomial coefficients a_0, a_1, a_2, ..., a_n
value of independent variable x_0
the degree of the polynomial *n*

STEP 1: initialize $value = a_n$
STEP 2: for *i* from $n-1$ downto 0
replace $value$ by $value \cdot x_0 + a_i$
OUTPUT: $value$

With this algorithm, polynomial evaluation can be carried out with n additions and n multiplications.

For Exercises 15 - 18, make use of the fact that when the sum of a convergent alternating series is approximated using the sum of the first n terms, the error in this approximation is smaller than the magnitude of the $(n+1)$-st term; *i.e.*, if $\sum(-1)^n a_n$ is an alternating series with sum S, then

$$\left| S - \sum_{k=0}^{n-1} (-1)^k a_k \right| < a_n.$$

15. The value of π is given by

$$\pi = 4\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + - \cdots\right).$$

(a) Construct an algorithm to approximate the value of π to within a specified tolerance, ϵ.

(b) Test your algorithm with a tolerance value of $\epsilon = 5 \times 10^{-3}$.

(a)

GIVEN: convergence parameter ϵ
maximum number of terms *Nmax*

STEP 1: initialize *sum* to 4 and *sign* to -1
STEP 2: for *i* from 1 to *Nmax*
STEP 3: set $term = 4/(2i+1)$
STEP 4: if $term < \epsilon$, OUTPUT *sum*
STEP 5: replace *sum* by $sum + sign * term$
STEP 6: replace *sign* by $-sign$
end
OUTPUT: "more terms needed"

(b) With a tolerance of 5×10^{-3}, it takes 400 terms to produce the estimate $\pi \approx 3.1391$, which is in error by roughly 2.5×10^{-3}.

16. The value of $1/e$ is given by

$$1/e = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + - \cdots.$$

(a) Construct an algorithm to approximate the value of $1/e$ to within a specified tolerance, ϵ.

(b) Test your algorithm with a tolerance value of $\epsilon = 5 \times 10^{-7}$.

(a)

GIVEN: convergence parameter ϵ
maximum number of terms *Nmax*

STEP 1: initialize *sum* to 1, *term* to 1 and *sign* to -1
STEP 2: for *i* from 1 to *Nmax*
STEP 3: replace $term$ by $term/i$
STEP 4: if $term < \epsilon$, OUTPUT *sum*
STEP 5: replace *sum* by $sum + sign * term$
STEP 6: replace *sign* by $-sign$
end
OUTPUT: "more terms needed"

(b) With a tolerance of 5×10^{-7}, ten terms are needed to produce the estimate $1/e \approx 0.367879189$, which is in error by roughly 2.5×10^{-7}.

17. The value of $\sin(\pi/10)$ is given by

$$\sin\left(\frac{\pi}{10}\right) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}\left(\frac{\pi}{10}\right)^{2n+1} = \frac{\pi}{10} - \frac{1}{3!}\left(\frac{\pi}{10}\right)^3 + \frac{1}{5!}\left(\frac{\pi}{10}\right)^5 - \frac{1}{7!}\left(\frac{\pi}{10}\right)^7 + - \cdots.$$

(a) Construct an algorithm to approximate the value of $\sin(\pi/10)$ to within a specified tolerance, ϵ.

(b) Test your algorithm with a tolerance value of $\epsilon = 5 \times 10^{-7}$. Note that the exact value of $\sin(\pi/10)$ is $\frac{1}{4}(\sqrt{5}-1)$.

(a)

GIVEN: convergence parameter ϵ
maximum number of terms *Nmax*

STEP 1: initialize *sum* to $\pi/10$, *term* to $\pi/10$ and *sign* to -1
STEP 2: for *i* from 1 to *Nmax*
STEP 3: replace $term$ by $term(\pi/10)^2/((2i)(2i+1))$
STEP 4: if $term < \epsilon$, OUTPUT *sum*
STEP 5: replace *sum* by $sum + sign * term$
STEP 6: replace *sign* by $-sign$
end
OUTPUT: "more terms needed"

(b) With a tolerance of 5×10^{-7}, three terms are needed to produce the estimate $\sin(\pi/10) \approx 0.309017054$, which is in error by roughly 6.0×10^{-8}.

18. The value of $\cos(\pi/5)$ is given by

$$\cos\left(\frac{\pi}{5}\right) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}\left(\frac{\pi}{5}\right)^{2n} = 1 - \frac{1}{2!}\left(\frac{\pi}{5}\right)^2 + \frac{1}{4!}\left(\frac{\pi}{5}\right)^4 - \frac{1}{6!}\left(\frac{\pi}{5}\right)^6 + - \cdots.$$

(a) Construct an algorithm to approximate the value of $\cos(\pi/5)$ to within a specified tolerance, ϵ.

(b) Test your algorithm with a tolerance value of $\epsilon = 5 \times 10^{-7}$. Note that the exact value of $\cos(\pi/5)$ is $\frac{1}{4}(\sqrt{5}+1)$.

(a)

GIVEN: convergence parameter ϵ
maximum number of terms *Nmax*

STEP 1: initialize *sum* to 1, *term* to 1 and *sign* to -1
STEP 2: for *i* from 1 to *Nmax*
STEP 3: replace $term$ by $term(\pi/5)^2/((2i-1)(2i))$
STEP 4: if $term < \epsilon$, OUTPUT *sum*
STEP 5: replace *sum* by $sum + sign * term$
STEP 6: replace *sign* by $-sign$
end
OUTPUT: "more terms needed"

(b) With a tolerance of 5×10^{-7}, five terms are needed to produce the estimate $\cos(\pi/5) \approx 0.809016997$, which is in error by roughly 2.6×10^{-9}.