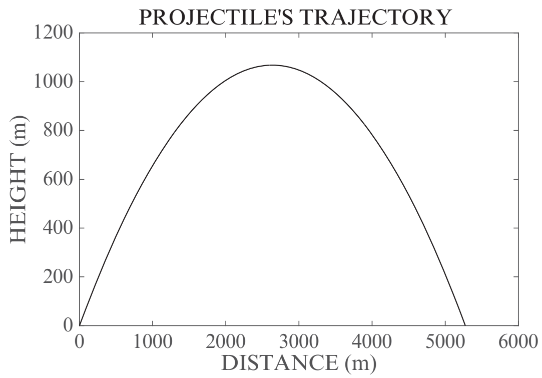
In addition, the following figure is created in the Figure Window:



7.13 PROBLEMS

1. Write a user-defined MATLAB function for the following math function:

$$y(x) = 0.6x^3e^{-0.47x} + 1.5x^2e^{-0.6x}$$

The input to the function is x and the output is y. Write the function such that x can be a vector (use element-by-element operations).
(*a*) Use the function to calculate $y(-2)$ and $y(4)$.
(*b*) Use the function to make a plot of the function $y(x)$ for $-4 \le x \le 8$.

2. Write a user-defined MATLAB function for the following math function:

$$r(\theta) = 3\sin(3\cos(0.5\theta))$$

The input to the function is θ (in radians) and the output is r. Write the function such that θ can be a vector.
(*a*) Use the function to calculate $r(\pi/6)$ and $r(5\pi/6)$.
(*b*) Use the function to plot (polar plot) $r(\theta)$ for $0 \le \theta \le 4\pi$.

3. In the U.S. fuel efficiency of cars is specified in miles per gallon (mpg). In Europe it is often expressed in liters per 100 km. Write a MATLAB user-defined function that converts fuel efficiency from mpg to liters per 100 km. For the function name and arguments, use `Lkm=mpgToLpkm(mpg)`. The input argument `mpg` is the efficiency in mi/gl, and the output argument `Lkm` is the efficiency in liters per 100 km (rounded to the nearest hundredth). Use the function in the Command Window to:
(*a*) Determine the fuel efficiency in liters per 100 km of a car whose fuel efficiency is 21 mi/gal.
(*b*) Determine the fuel efficiency in liters per 100 km of a car whose fuel efficiency is 36 mi/gal.

4. Pressure in U.S. customary units is measured in psi (pound per square inch). In SI metric units pressure is measured in Pa (N/m^2). Write a user-defined MATLAB function that converts pressure given in units of psi to pressure in units of Pa. For the function name and arguments, use `[Pa] = PsiToPa(psi)`. The input argument `psi` is the pressure in units of psi to be converted, and the output argument `Pa` is the converted pressure in units of Pa (rounded to the nearest integer). Use the function in the Command Window to:
(*a*) Convert 120 psi to units of Pa.
(*b*) Convert 3,000 psi to units of Pa.

5. Tables of material properties list density, in units of kg/m^3, when the international system of units (SI) is used, and list specific weight, in units of lb/in.^{3}, when the U.S. customary system of units is used. Write a user-defined MATLAB function that converts density to specific weight. For the function name and arguments, use `[sw] = DenToSw(den)`. The input argument `den` is the density of a material in kg/m^3, and the output argument `sw` is the specific weight in lb/in.^{3}. Use the function in the Command Window to:
(*a*) Determine the specific weight of copper whose density is 8,960 kg/m^3.
(*b*) Determine the specific weight of concrete whose density is 2,340 kg/m^3.

6. Write a user-defined MATLAB function that converts torque given in units of N-m to torque in units of lb-ft. For the function name and arguments, use `lbft = NmTOlbft(Nm)`. The input argument `Nm` is the torque in N-m, and the output argument `lbft` is the torque in lb-ft (rounded to the nearest integer). Use the function to convert 2,000 N-m to units of lb-ft.

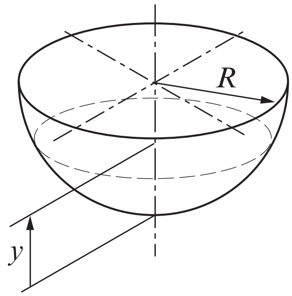
7. The body surface area (*BSA*) in m^2 of a person (used for determining dosage of medications) can be calculated by the formula (Mosteller formula):

$$BSA = \sqrt{H \times W / 3131}$$

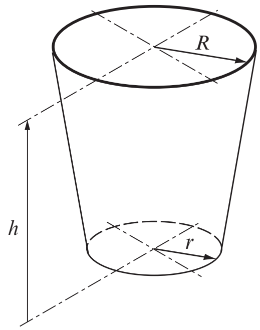
in which H is the person's height in inches, and W is the persons weight in lb.
Write a MATLAB user-defined function that calculates the body surface area. For the function name and arguments, use `BSA = BodySurA(w,h)`. The input arguments `w` and `h` are the weight and height, respectively. The output argument `BSA` is the *BSA* value. Use the function to calculate the body surface area of:
(*a*) A 170-lb, 5-ft 10-in. tall person.
(*b*) A 220-lb, 6-ft 5-in. tall person.

8. The fuel tank shown in the figure in shaped as a half a sphere with $R = 24$ in.

   Write a user-defined function that calculates the volume of fuel in the tank (in gallons) as a function of the height y (measured from the bottom). For the function name and arguments, use `V = Volfuel(y)`. Use the function to make a plot of the volume as a function of y for $0 \le y \le 24$ in.



9. A paper cup is designed to have a geometry of a frustum of a cone. Write a user-defined function that determines the volume and the surface area (side plus bottom) of the cup for given values of r, R, and h. For the function name and arguments, use `[V, S] = VolSArea(r,R,h)`. The input arguments `r`, `R`, and `h` are the radius of the base, the radius of the top and the height, respectively (all in units of inches). The output arguments `V` and `S` are the volume (in units of U.S. fluid ounce) and the surface area (in units of in.^{2}), respectively. Use the function to determine the volume and the surface area of cups with the following dimensions:



   (*a*) $r = 2$ in., $R = 3.5$ in., $h = 4.25$ in.
   (*b*) $r = 2.5$in., $R = 3.5$ in., $h = 4.5$ in.

10. The relative humidity, RH, at sea level can be calculated from measured values of the dry-bulb temperature, T_{db}, and the wet-bulb temperature T_{wb} by (temperatures in degrees Celsius):

$$RH = \frac{VP}{SVP}100$$

where VP is the vapor pressure given by:

$$VP = e^{\frac{16.78T_{wb}-116.9}{T_{wb}+237.3}} - 0.066858(1+0.00115T_{wb})(T_{db}-T_{wb})$$

and SVP is the saturated vapor pressure given by:

$$SVP = e^{\frac{16.78T_{db}-116.9}{T_{db}+237.3}}$$

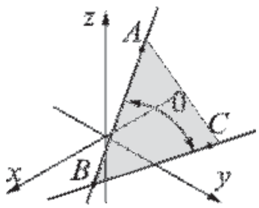
Write a user-defined function for calculating RH for given T_{db} and T_{wb}. For the function name and arguments, use `RH = RelHum(Tdb,Twb)`. The input arguments are `Tdb` and `Twb` are the dry-bulb and wet-buld temperatures, respectively in °F. The output argument `RH` is the relative humidity in percent (rounded to the nearest integer). Inside the user-defined function use a subfunction, or an anonymous function to convert the unit of the temperature from Cesius to Fahrenheit. Use the function to determine the relative humidity for the following conditions:

(*a*) $T_{db} = 75°$F, $T_{wb} = 69°$F. (*b*) $T_{db} = 93°$F, $T_{wb} = 90°$ F.

11. Write a user-defined function that calculates grade point average (GPA) on a scale of 0 to 5, where $A = 5$, $B = 4$, $C = 3$, $D = 2$, and $F = 0$. For the function name and arguments, use `GPA = GradePtAve(G,C)`. The input argument `G` is a vector whose elements are letter grades *A*, *B*, *C*, *D*, or *F* entered as a string (e.g., `['ABACFB']`). The input argument `C` is a vector with the corresponding credit hours. The output argument `GPA` is the calculated GPA rounded to the nearest tenth (i.e., 3.75 is rounded to 3.8, and 3.749 is rounded to 3.7). Use the function to calculate the GPA for a student with the following record:

Grade	*A*	*B*	*F*	*C*	*B*	*A*	*D*	*A*
Credit Hours	4	3	3	2	3	4	3	3

12. Write a user-defined MATLAB function that determines the angle that forms by the intersection of two lines. For the function name and arguments, use `th=anglines(A,B,C)`. The input arguments to the function are vectors with the coordinates of the points *A*, *B*, and *C*, as shown in the figure, which can be two- or three-dimensional. The output `th` is the angle in degrees. Use the function `anglines` for determining the angle for the following cases:



(*a*) $A(-5, -1, 6)$, $B(2.5, 1.5, -3.5)$, $C(-2.3, 8, 1)$
(*b*) $A(-5.5, 0)$, $B(3.5, -6.5)$, $C(0, 7)$

13 Write a user-defined MATLAB function that determines the time elapsed between two events during a day. For the function name and arguments, use `dt = timediff(TA,ap1,TB,ap2)`. The input arguments to the function are:
`TA` is a two-element vector with the time of the first event. The first element is the hour and the second element is the minute.
`ap1` is a string `'AM'` or `'PM'` which corresponds to the time of the first event.
`TB` is a two-element vector with the time of the second event. The first element is the hour and the second element is the minute.
`ap2` is a string `'AM'` or `'PM'` which corresponds to the time of the second event.

The output argument `dt` is a two-element vector with the time elapsed between two events. The first element is the number of hours and the second element is number of minutes.

The function displays an error message if the time entered for event *B* is before the time entered for event *A*.

Use the function to determine the time elapsed between the following events:

(*a*) Event A: 5:37 AM; Event B: 2:51 PM.
(*b*) Event A: 12:53 PM; Event B: 6:12 PM.
(*c*) Event A: 11:32 PM; Event B: 3:18 PM. (Error situation.)

14. Write a user-defined MATLAB function that determines the unit vector in the direction of the line that connects two points (A and B) in space. For the function name and arguments, use `n = unitvec(A,B)`. The input to the function are two vectors `A` and `B`, each with the Cartesian coordinates of the corresponding point. The output `n` is a vector with the components of the unit vector in the direction from A to B. If points A and B have two coordinates each (they are in the x y plane), then `n` is a two-element vector. If points A and B have three coordinate each (general points in space), then `n` is a three-element vector. Use the function to determine the following unit vectors:
(*a*) In the direction from point (–0.7, 2.1) to point (9, 18).
(*b*) In the direction from point (10, –3.5, –2.5) to point (–11, 6.5, 5.9).

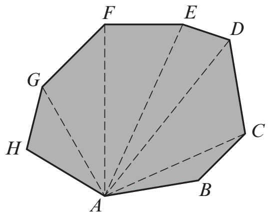
15. Write a user-defined MATLAB function that determines the cross product of two vectors. For the function name and arguments, use `w=crosspro(u,v)`. The input arguments to the function are the two vectors, which can be two- or three-dimensional. The output `w` is the result (a vector). Use the function `crosspro` for determining the cross product of:
(*a*) Vectors $a = 3i + 11j$ and $b = 14i - 7.3j$.
(*b*) Vectors $c = -6i + 14.2j + 3k$ and $d = 6.3i - 8j - 5.6k$.

16. The area of a triangle ABC can be calculated by:

$$A = \frac{1}{2}|\boldsymbol{AB} \times \boldsymbol{AC}|$$

where ***AB*** is the vector from vertex A to vertex B and ***AC*** is the vector from vertex A to vertex C. Write a user-defined MATLAB function that determines the area of a triangle given its vertices' coordinates. For the function name and arguments, use `[Area] = TriArea(A,B,C)`. The input arguments `A`, `B`, and `C` are vectors, each with the coordinates of the corresponding vertex. Write the code of `TriArea` such that it has two subfunctions—one that determines the vectors ***AB*** and ***AC*** and another that executes the cross product. (If available, use the user-defined functions from Problem 15). The function should work for a triangle in the x-y plane (each vertex is defined by two coordinates) or for a triangle in space (each vertex is defined by three coordinates). Use the function to determine the areas of triangles with the following vertices:
(*a*) $A = (1, 2)$, $B = (10, 3)$, $C = (6, 11)$
(*b*) $A = (-1.5, -4.2, -3)$, $B = (-5.1, 6.3, 2)$, $C = (12.1, 0, -1.5)$

17. As shown in the figure, the area of a convex polygon can be calculated by adding the area of the triangles that the polygon can be divided into. Write a user-defined MATLAB function that calculates the area of a convex *n*-sided polygon. For the function name and arguments, use `A = APolygon(Crd)`. The input argument `Crd` is a two-column matrix where each row contains the coordinates of a vertex (first column is the *x* coordinate and the second column is the *y* coordinate). The vertices are listed in the order that they are connected to form the polygon (i.e., coordinates of point *A* in the first row, point *B* in the second, and so on). The output argument `A` is the area of the polygon. Write the code of `APolygon` such that it has a subfunction that calculates the area of a triangle for given vertices' coordinates. Use `APolygon` to calculate the area of the polygon shown in the figure. The coordinates of the vertices are: *A*(1, 1), *B*(7, 2), *C*(10, 5), *D*(9, 11), *E*(6, 12), *F*(1, 12), *G*(–3, 8), *H*(–4, 4).

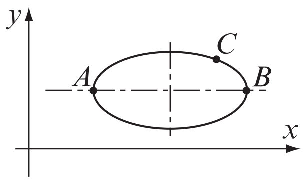


18. Write a user-defined function that determines the location of the center and the radius of a circle that passes through three given points in a plane. The function also creates a plot that shows the circle and the points. For the function name and arguments, use `[C R]=Circle3Pts(A,B,C)`. The input arguments `A`, `B`, and `C` are each a two-element vector with the *x* and *y* coordinates of the corresponding point. The output argument `C`, is a vector with the coordinates of the center the output argument `R`, is the radius (both rounded to the nearest hundredth). Use the function with the following three points: *A*(7, 1.2), *B*(0.5, 2.6), and *C*(–2.4, –1.4).

19. Write a user-defined MATLAB function that converts integers written in decimal form to binary form. Name the function `b=Bina(d)`, where the input argument `d` is the integer to be converted and the output argument `b` is a vector with 1s and 0s that represents the number in binary form. The largest number that could be converted with the function should be a binary number with 16 1s. If a larger number is entered as `d`, the function should display an error message. Use the function to convert the following numbers:
(*a*) 100 (*b*) 1,002 (*c*) 52,601 (*d*) 200,090

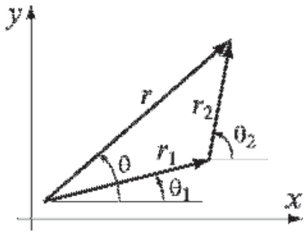
20. Write a user-defined function that plots a triangle and the circle that is inscribed inside, given the coordinates of its vertices. For the function name and arguments, use `TriCirc(A,B,C)`. The input arguments are vectors with the *x* and *y* coordinates of the vertices, respectively. This function has no output arguments. Use the function with the points (2.6, 3.2), (11, 14.5), and (–2, 2.8).

21. Write a user-defined function that plots an ellipse with axes that are parallel to the x and y axes, given the coordinates of its vertices and the coordinates of another point that the ellipse passes through. For the function name and arguments, use `ellipseplot(A,B,C)`. The input arguments `A` and `B` are each a two-element vector with the coordinates of the vertices, and `C` is a two-element vector with the coordinates of another point on the ellipse (see figure), respectively. This function has no output arguments. Use the function to plot the following ellipses:



(*a*) $A(2,3)$, $B(11,3)$, $C(10,4)$ (*b*) $A(2,11)$, $B(2,-4)$, $C(4,8)$

22. In polar coordinates a two-dimensional vector is given by its radius and angle (r, θ). Write a user-defined MATLAB function that adds two vectors that are given in polar coordinates. For the function name and arguments, use `[r th] = AddVecPol(r1,th1,r2,th2)`, where the input arguments are (r_1, θ_1) and (r_2, θ_2), and the output arguments are the radius and angle of the result. Use the function to carry out the following additions:

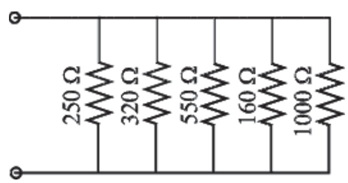


(*a*) $r_1 = (5, 23°)$, $r_2 = (12, 40°)$ (*b*) $r_1 = (6, 80°)$, $r_2 = (15, 125°)$

23. Write a user-defined function that determines if a number is a prime number. Name the function `pr=Trueprime(m)`, where the input arguments `m` is a positive integer and the output argument `pr` is 1 if `m` is a prime number and 0 if `m` is not a prime number. Do not use MATLAB's built-in functions `primes` and `isprime`. If a negative number or a number that is not an integer is entered when the function is called, the error message "The input argument must be a positive integer." is displayed.
(*a*) Use the function with 733, 2001, and 107.5.
(*b*) Write a MATLAB program in a script file that makes use of `Trueprime` and finds the smallest prime number that remains a prime number when added to its reverse (37 is the reverse of 73).

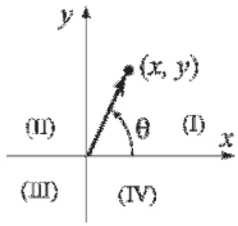
24. The harmonic mean H of a set of n positive numbers $x_1, x_2, \ldots, x_n$ is defined by:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}}$$



Write a user-defined function that calculates the harmonic mean of a set of numbers. For function name and arguments use `G=Harmean(x)`, where the input argument x is a vector of numbers (any length) and the output argument H is their harmonic mean. In electrical engineering the equivalent resistance of resistors connected in parallel is equal to the harmonic mean of the values of the resistors divided by the number of the resistors. Use the user-defined function Harmean to calculate the equivalent resistance of the resistors shown in the figure.

25. Write a user-defined function that determines the polar coordinates of a point from the Cartesian coordinates in a two-dimensional plane. For the function name and arguments, use [th rad]=CartToPolar(x,y). The input arguments are the x and y coordinates of the point, and the output arguments are the angle θ and the radial distance to the point. The angle θ is in degrees and is measured relative to the positive x axis, such that it is a positive number in quadrants I and II, and a negative number in quadrant III and IV. Use the function to determine the polar coordinates of points (14, 9), (–11, –20), (–15, 4), and (13.5, –23.5).



26. Write a user-defined function that determines the value that occurs most often in a set of data that is given in a two-dimensional matrix. For the function name and arguments, use [v, q] =matrixmode(x). The input argument x is a $m \times n$ matrix of any size with numerical values, and the output arguments v and q are the values that occur most often and the number of times they occur. If there are two, or more, values that occur most often than v is a vector with these values. Do not use the MATLAB built-in function mode. Test the function three times. For input create a 5×6 matrix using the following command: x=randi(10,5,6).

27. Write a user-defined function that sorts the elements of a vector from the largest to the smallest. For the function name and arguments, use y=downsort(x). The input to the function is a vector x of any length, and the output y is a vector in which the elements of x are arranged in a descending order. Do not use the MATLAB built-in functions sort, max, or min. Test your function on a vector with 14 numbers (integers) randomly distributed between –30 and 30. Use the MATLAB randi function to generate the initial vector.

28. Write a user-defined function that sorts the elements of a matrix. For the function name and arguments, use B = matrixsort(A), where A is any size $(m \times n)$ matrix and B is a matrix of the same size with the elements of A rearranged in descending order column after column with the (1,1) element the largest and the (m,n) element the smallest. If available, use the user-defined function downsort from the previous problem as a subfunction

within `matrixsort`.

Test your function on a 4×7 matrix with elements (integers) randomly distributed between –30 and 30. Use MATLAB's `randi` function to generate the initial matrix.

29. Write a user-defined MATLAB function that finds the largest element of a matrix. For the function name and arguments, use `[Em,rc] = matrixmax(A)`, where `A` is any size matrix. The output argument `Em` is the value of the largest element, and `rc` is a two-element vector with the address of the largest element (row and column numbers). If there are two, or more, elements that have the maximum value, the output argument `rc` is a two-column matrix where the rows list the addresses of the elements. Test the function three times. For input create a 4×6 matrix using the following command: `x=randi([-20 100],4,6)`.

30. Write a user-defined MATLAB function that calculates the determinant of a 3×3 matrix by using the formula:

$$det = A_{11}\begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} - A_{12}\begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} \end{vmatrix} + A_{13}\begin{vmatrix} A_{21} & A_{22} \\ A_{31} & A_{32} \end{vmatrix}$$

For the function name and arguments, use `d3 = det3by3(A)`, where the input argument `A` is the matrix and the output argument `d3` is the value of the determinant. Write the code of `det3by3` such that it has a subfunction that calculates the 2×2 determinant. Use `det3by3` for calculating the determinants of:

(*a*) $\begin{bmatrix} 1 & 3 & 2 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix}$ (*b*) $\begin{bmatrix} -2.5 & 7 & 1 \\ 5 & -3 & -2.6 \\ 4 & 2 & -1 \end{bmatrix}$

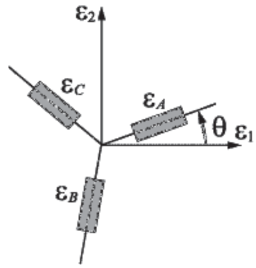
31. The shortest distance between two points on the surface of the globe (great-circle distance) can be calculated by using the haversine formula. If Φ_1 and λ_1 are the latitude and longitude of point 1 and Φ_2 and λ_2 are the latitude and longitude of point 2, the great circle distance between the points is given by:

$$d = 2R\sin^{-1}\left(\sqrt{a}\right)$$

where $a = \sin^2\left(\frac{\Phi_2 - \Phi_1}{2}\right) + \cos\Phi_1 \cos\Phi_2 \cos\Phi_1 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$, and $R = 3,959$ mi is the Earth radius. Write a user-defined function that determines the distance between two points on the Earth. For the function name and arguments, use `dis = GreatCirDis(Lat1,Lng1,Lat2,Lng2)`, where the input arguments are the latitude and longitude of the two points (degrees in decimal format), and `dis` is the great-circle distance in miles. Use the function to calculate the distance between London (51.50853°, –0.12574°) and New

York City (40.71427°, –74.00597°).

32. Delta rosette is a set of three strain gages oriented at 120° relative to each other. The strain measured with each of the strain gages is ε_A, ε_B, and ε_C. The principal strains ε_1 and ε_2 can be calculated from the strains measured with the rosette by:



$$\epsilon_{1,2} = \frac{\epsilon_A + \epsilon_B + \epsilon_C}{3} \pm \frac{\sqrt{2}}{3}\sqrt{(\epsilon_A - \epsilon_B)^2 + (\epsilon_B - \epsilon_C)^2 + (\epsilon_C - \epsilon_A)^2}$$

Write a user-defined MATLAB function that determines the principal strains given the strains ε_A, ε_B, and ε_C. For the function name and arguments, use `[P1, P2]=DeltaRos(A,B,C)`. The input arguments `A`, `B`, and `C` are the values of the three strains measured by the rosette. The output arguments `P1` and `P2` are the values of the principal strains.

Use the function to determine the principal strains for the following cases:

(*a*) $\epsilon_A = 42\ \mu$, $\epsilon_B = 970\ \mu$, $\epsilon_C = 340\ \mu$.

(*b*) $\epsilon_A = 110\ \mu$, $\epsilon_B = 80\ \mu$, $\epsilon_C = -60\ \mu$.

33. In a lottery the player has to select several numbers out of a list. Write a user-defined function that generates a list of n integers that are uniformly distributed between the numbers a and b. All the selected numbers on the list must be different. For function name and arguments, use `x=lotto(a,b,n)` where the input arguments are the numbers a and b, and n, respectively. The output argument `x` is a vector with the selected numbers.

(*a*) Use the function to generate a list of seven numbers from the numbers 1 through 59.

(*b*) Use the function to generate a list of eight numbers from the numbers 50 through 65.

(*c*) Use the function to generate a list of nine numbers from the numbers –25 through –2.

34. The Taylor series expansion for $\sin x$ about $x = 0$ is given by:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^{57}}{7!} + \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

where x is in radians. Write a user-defined function that determines $\sin x$ using Taylor's series expansion. For function name and arguments, use `y=sinTay(x)`, where the input argument `x` is the angle in degrees and the output argument `y` is the value of $\sin x$. Inside the user-defined function, use a loop for adding the terms of the Taylor series. If a_n is the nth term in the series, then the sum S_n of the n terms is $S_n = S_{n-1} + a_n$. In each pass, calculate the estimated error E given by $E = \left|\frac{S_n - S_{n-1}}{S_{n-1}}\right|$. Stop adding terms when

$E \le 0.000001$. Since $\sin\theta = \sin(\theta \pm 360n)$ (n is an integer) write the user-defined function such that if the angle is larger than 360°, or smaller than –360°, then the Taylor series will be calculated using the smallest number of terms (using a value for x that is closest to 0).

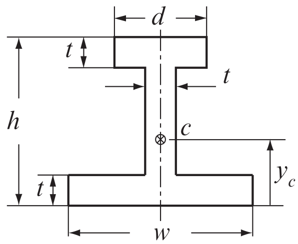
Use `sinTay` for calculating:

(*a*) $\sin 39°$ (*b*) $\sin 205°$ (*c*) $\sin(-70°)$.

(*d*) $\sin 754°$ (*e*) $\sin 19{,}000°$ (*f*) $\sin(-748°)$

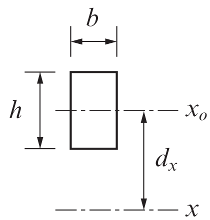
Compare the values calculated using `sinTay` with the values obtained by using MATLAB's built-in `sind` function.

35. Write a user-defined function that determines the coordinate y_c of the centroid of the I-shaped cross-sectional area shown in the figure. For the function name and arguments, use `yc = centroidI(w,h,d,t)`, where the input arguments `w`, `h`, `d`, and `t` are the dimensions shown in the figure and the output argument `yc` is the coordinate y_c.

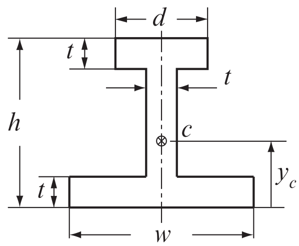


Use the function to determine y_c for a beam with w = 10 in., h = 8 in., d = 6 in., and t = 0.5 in.

36. The area moment of inertia I_{x_o} of a rectangle about the axis x_o passing through its centroid is $I_{x_o} = \frac{1}{12}bh^3$. The moment of inertia about an axis x that is parallel to x_o is given by $I_x = I_{x_o} + Ad_x^2$, where A is the area of the rectangle, and d_x is the distance between the two axes.



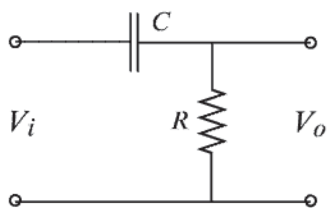
Write a MATLAB user-defined function that determines the area moment of inertia I_{x_c} of a I-beam about the axis that passes through its centroid (see drawing). For the function name and arguments use `Ixc=IxcBeam(w,h,d,t)`, where the input arguments `w`, `h`, `d`, and `t` are the dimensions shown in the figure and the output argument `Ixc` is I_{x_c}. For finding the coordinate y_c of the centroid, use the user-defined function `centroidI` from the previous problem as a subfunction inside `IxcBeam`.



(The moment of inertia of a composite area is obtained by dividing the area into parts and adding the moments of inertia of the parts.)

Use the function to determine the moment of inertia for a beam with w = 10 in., h = 8 in., d = 6 in., and t = 0.5 in.

37. The simple *RC* high-pass filter shown in the figure passes signals with frequencies higher than a certain cutoff frequency. The ratio of the magnitudes of the voltages is given by:



$$RV = \left|\frac{V_o}{V_i}\right| = \frac{\omega RC}{\sqrt{1+\omega^2 R^2 C^2}}$$

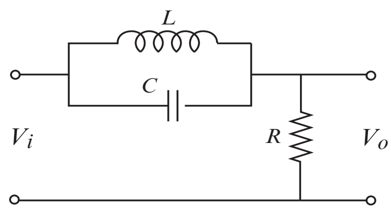
where $\omega = 2\pi f$, and *f* is the frequency of the input signal.

Write a user-defined MATLAB function that calculates the ratio of magnitudes for given values of *R*, *C*, and *f*. For the function name and arguments, use `RV = RCFilt(R,C,f)`. The input arguments are `R`, the size of the resistor in Ω (ohms); `C`, the size of the capacitor in F (farad); and `f`, the frequency of the input signal in Hz (hertz). Write the function such that `f` can be a vector.

Write a program in a script file that uses the `RCFilt` function to generate a plot of *RV* as a function of *f* for $10 \le f \le 100,000$ Hz. The plot has a logarithmic scale on the horizontal axis. When executed, the script file asks the user to enter the values of *R* and *C*. Label the axes of the plot.

Run the script file with $R = 80$ Ω, and $C = 5$ μF.

38. A circuit that filters out a certain frequency is shown in the figure. In this filter, the ratio of the magnitudes of the voltages is given by:



$$RV = \left|\frac{V_o}{V_i}\right| = \frac{\left|R(1-\omega^2 LC)\right|}{\sqrt{(\omega L)^2 + (R - R\omega^2 LC)^2}}$$

where $\omega = 2\pi f$, and *f* is the frequency of the input signal.

Write a user-defined MATLAB function that calculates the ratio of magnitudes. For the function name and arguments, use `RV=filtafreq(R,C,L,f)`. The input arguments are `R` the size of the resistor in Ω (ohms); `C`, the size of the capacitor in F (farad); `L`, the inductance of the coil in H (henrys); and `f`, the frequency of the input signal in Hz (hertz). Write the function such that `f` can be a vector.

Write a program in a script file that uses the `filtafreq` function to generate a figure with two plots of *RV* as a function of *f* for $10 \le f \le 100,000$ Hz. In one plot $C = 160$ μF, $L = 45$ mH, and $R = 200$ Ω, and in the second plot *C* and *L* are unchanged but $R = 50$ Ω. The plot has a logarithmic scale on the horizontal axis. Label the axes and display a legend.

39. The first derivative $\frac{df(x)}{dx}$ of a function $f(x)$ at a point $x = x_0$ can be approximated with the four-point central difference formula:

$$\frac{df}{dx} = \frac{f(x_0-2h)-8f(x_0-h)+8f(x_0+h)-f(x_0+2h)}{12h}$$

where h is a small number relative to x_0. Write a user-defined function function (see Section 7.9) that calculates the derivative of a math function $f(x)$ by using the four-point central difference formula. For the user-defined function name, use `dfdx=FoPtder(Fun,x0)`, where `Fun` is a name for the function that is passed into `FoPtder`, and `x0` is the point where the derivative is calculated. Use $h = x_0/100$ in the four-point central difference formula. Use the user-defined function `FoPtder` to calculate the following:

(*a*) The derivative of $f(x) = x^3e^{2x}$ at $x_0 = 0.6$.

(*b*) The derivative of $f(x) = \frac{3^x}{x^2}$ at $x_0 = 2.5$.

In both cases compare the answer obtained from `FoPtder` with the analytical solution (use `format long`).

40. In lottery the player has to guess correctly r numbers that are drawn out of n numbers. The probability, P, of guessing m numbers out of the r numbers can be calculated by the expression:

$$P = \frac{C_{r,m}C_{(n-r),(r-m)}}{C_{n,r}}$$

where $C_{x,y} = \frac{x!}{y!(x-y)!}$. Write a user-defined MATLAB function that calculates P. For the function name and arguments, use `P = ProbLottery(m,r,n)`. The input arguments are `m`, the number of correct guesses; `r`, the number of numbers that need to be guessed; and `n`, the number of numbers available. Use a subfunction inside `ProbLottery` for calculating $C_{x,y}$.

(*a*) Use `ProbLottery` for calculating the probability of correctly selecting 3 of 6 numbers that are drawn out of 49 numbers in a lottery game.

(*b*) Consider a lottery game in which 6 numbers are drawn out of 49 numbers. Write a program in a script file that displays a table with seven rows and two columns. The first column has the numbers 0, 1, 2, 3, 4, 5, and 6, which are the number of numbers guessed correctly. The second column show the corresponding probability of making the guess.