**Chapter 2: Tutorial Lessons- Lesson 1**

|  |  |
| --- | --- |
| **Objective :** | 1. Arithmetic operations: Compute the following quantities:   Area = pi \* r2  Where r =  2. Exponential and logarithms: Solve 3x = 17 for x.  3. Trigonometry: *y* = cosh2 *x −* sinh2 *x*, with *x* = 32*π*.  4. Complex Numbers: Check for Euler’s Formula by comparing LHS and RHS. |
| **MATLAB**  **Code:** | % 1. Arithmatic operators Area program  radius = pi^(1/3) - 1  AREA = pi \* radius^2  % 2. Exp and log  x = log(17) / log(3)  % 3. Trigo (y = cosh^2 x − sinh^2 x)  x = 32\*pi;  y = (cosh(x) )^2 - (sinh(x) )^2  % 4. Complex nos (Check the Euler’s Formula)  x = (pi/4)  res\_1 = exp(j\*x)  res\_2 = cos(x) + j\*sin(x)  if res\_1 == res\_2  disp('verified euler formula')  end |
| **Output:** | 1. radius =  0.4646  AREA =  0.6781  2. x =  2.5789  3. y =  0  4. x = 0.7854  res\_1 =  0.7071 + 0.7071i  res\_2 =  0.7071 + 0.7071i  verified euler formula |

**Lesson 2: Creating and Working with Arrays**

**of Numbers**

|  |  |
| --- | --- |
| **Objective :** | 1. The **equation of a straight line** is y = mx+c where m and c are constants. Compute the y-coordinates of a line with slope m = 0.5 and the intercept c = −2 at the following x-coordinates: (0, 1.5, 3, 4, 5, 7, 9, and 10) . 2. **Multiply, divide, and exponentiate vectors**: Create a vector t with 10 elements: 1, 2, 3, . . ., 10. Now compute the following quantities:   • x = t sin(t).  • y = t−1  t+1.  • z = sin(t2)  t2   1. **Points on a circle:** All points with coordinates x = r cos θ and y = r sin θ,   where r is a constant, lie on a circle with radius r, i.e., they satisfy the  equation x2 + y2 = r2. Create a column vector for θ with the values 0, π/4,  π/2, 3π/4, π, and 5π/4.  Take r = 2 and compute the column vectors x and y. Now check that x and y  indeed satisfy the equation of circle, by computing the radius r = **.**   1. **The geometric series:** This is funky! You know how to compute xn   element-by-element for a vector x and a scalar exponent n. How about computing  nx, and what does it mean? The result is again a vector with elements  nx1, nx2, nx3 etc.  Now take the sum of this vector with the command s = sum(x). Calculate the limit and compare the computed sum s. Repeat the procedure taking n from 0 to 50 and then from 0 to 100. |
| **MATLAB**  **Code:** | disp('Equation of a straight line:')      x = [0, 1.5, 3 ,4 ,5, 7, 9 , 10];      m = 0.5;      c = -2;      y = m.\*x + c      disp('Multiply, divide, and exponentiate vectors: ')      t = 1:10;      x = t.\*sin(t)      y = (t-1) ./ (t+1)      z = sin( t.^2  ) ./ t.^2  disp('Points on a circle')      RADIUS = 2;      theta = 0:(pi/4):(5\*pi/4) ;      x = cos(theta)\*RADIUS      y = RADIUS\*sin(theta)      condition = sqrt(x.^2 + y.^2);      if (condition == RADIUS)           disp('Equation of circle is verified')      end    disp('Geometric series')      n = 0:1:10;      r = 0.5;      x = r.^n ;      s = sum(x);      result\_by\_formula = 1 / (1-r);      deviation\_1 = s - result\_by\_formula      n = 0:1:50;      x = r.^n ;      s = sum(x);      deviation\_2 = s - result\_by\_formula      n = 0:1:100;      x = r.^n ;      s = sum(x);      deviation\_3 = s - result\_by\_formula |
| **Output:** | Equation of a straight line:  y =  Columns 1 through 6  -2.0000 -1.2500 -0.5000 0 0.5000 1.5000  Columns 7 through 8  2.5000 3.0000  Multiply, divide, and exponentiate vectors:  x =  Columns 1 through 6  0.8415 1.8186 0.4234 -3.0272 -4.7946 -1.6765  Columns 7 through 10  4.5989 7.9149 3.7091 -5.4402  y =  Columns 1 through 6  0 0.3333 0.5000 0.6000 0.6667 0.7143  Columns 7 through 10  0.7500 0.7778 0.8000 0.8182  z =  Columns 1 through 6  0.8415 -0.1892 0.0458 -0.0180 -0.0053 -0.0275  Columns 7 through 10  -0.0195 0.0144 -0.0078 -0.0051  Points on a circle  x =  2.0000 1.4142 0.0000 -1.4142 -2.0000 -1.4142  y =  0 1.4142 2.0000 1.4142 0.0000 -1.4142  Equation of circle is verified  Geometric series  deviation\_1 = -9.7656e-04  deviation\_2 = -8.8818e-16  deviation\_3 = 0 |

**Lesson 3: Creating and Printing Simple Plots**

|  |  |
| --- | --- |
| **Objective :** | 1. A simple sine plot: Plot y = sin x, 0 ≤ x ≤ 2π, taking 100 linearly spaced points in the given interval. Label the axes and put ‘Plot created by yourname’ in the title. 2. An exponentially decaying sine plot: Plot y = e−0.4x sin x, 0 ≤ x ≤4π, taking 10, 50, and 100 points in the interval. [Be careful about computing. You need array multiplication between exp(-0.4\*x) and sin(x).] |
| **MATLAB**  **Code:** | disp('a simple sine plot')  theta = linspace(0, 2\*pi, 100);  y = sin(theta);  ylabel('Y')  xlabel('theta')  title('plot created by Dhruva')  plot(theta, y, 'O')  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  disp('An exponentially decaying sine curve')  n = input('10, 50 or 100 points? ');  switch n  case 10  disp('using 10 points')  theta = linspace(0, 4\*pi, 10);  y\_sin = sin(theta);  expo = exp( theta.\*(-0.4) ) ;  y = expo.\*y\_sin;  plot(theta, y)  title('using 10 points ')  ylabel('Y')  xlabel('theta')  case 50  disp('using 50 points')  theta = linspace(0, 4\*pi, 50);  y\_sin = sin(theta);  expo = exp( theta.\*(-0.4) ) ;  y = expo.\*y\_sin;  plot(theta, y)  title('using 50 points ')  ylabel('Y')  xlabel('theta')  case 100  disp('using 100 points')  theta = linspace(0, 4\*pi, 100);  y\_sin = sin(theta);  expo = exp( theta.\*(-0.4) ) ;  y = expo.\*y\_sin;  plot(theta, y)  title('using 100 points ')  ylabel('Y')  xlabel('theta')  end  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% |
| **Output:** | 1. Simple Sine plot with ‘O’s.  1. Decaying sine plots with 10, 50 and 100 points respectively: |

**Lesson 5: Creating and Executing a Function File**

|  |  |
| --- | --- |
| **Objective :** | **Convert temperature:** Write a function that outputs a conversion-table  for Celsius and Fahrenheit temperatures. The input of the function should be  two numbers: Ti and Tf , specifying the lower and upper range of the table in  Celsius. The output should be a two column matrix: the first column showing  the temperature in Celsius from Ti to Tf in the increments of 1oC and the  second column showing the corresponding temperatures in Fahrenheit. To  do this:  (i) create a column vector C from Ti to Tf with the command C = [Ti:Tf]’,  (ii) calculate the corresponding numbers in Fahrenheit using the formula [F = 9/5C + 32], and  (iii) make the final matrix with the command temp = [C F];. Note that your output will be named temp. |
| **MATLAB**  **Code:** | function [C F] = my\_temp\_conv(Ti, Tf)  %Enter the lower and upper limit respectively  % and this function shall return a tabular conversion between those  %values in differences of 1  %  % my\_temp\_conv (float, float)  %  celcius = Ti:1:Tf ;  farhenite = (9/5)\*celcius + 32 ;  C = celcius;  F = farhenite; |
| **Output:** | >> [C F] = my\_temp\_conv(25,30)  C =  25 26 27 28 29 30  F =  77.0000 78.8000 80.6000 82.4000 84.2000 86.0000 |

**Lesson 2**

|  |  |
| --- | --- |
| **Objective :** |  |
| **MATLAB**  **Code:** |  |
| **Output:** |  |

**Lesson 2**

|  |  |
| --- | --- |
| **Objective :** |  |
| **MATLAB**  **Code:** |  |
| **Output:** |  |