Shivek Khurana

Matlab Tut 2

11 February 2013

1. Write MATLAB function **sigma = ascsum(x)** that takes a one-dimensional array **x** of real

numbers and computes their sum **sigma** in the ascending order of magnitudes.

**Hint:** You may wish to use MATLAB functions **sort**, **sum**, and **abs**.

Ans 1 :

function sigma = ascsum(x)

sigma = sum(sort(abs(x)));

end

2. In this exercise you are to write MATLAB function **d = dsc(c)** that takes a one-dimensional array of numbers **c** and returns an array **d** consisting of all numbers in the array **c** with all neighboring duplicated numbers being removed. For instance, if **c = [1 2 2 2 3 1]**, then

**d = [1 2 3 1]**.

Ans 2 :

function d = dsc(c)

d = c(logical(diff([0 c])));

end

3. Write MATLAB function **p = fact(n)** that takes a nonnegative integer **n** and returns value of the factorial function n! = 1\*2\* … \*n. Add an error message to your code that will be executed when the input parameter is a negative number.

Ans 3 :

function f = fact(n)

f = 1;

if n >= 0

f = f\*n;

n = n-1;

if(n>0)

f = f\*fact(n);

end

else

error('Factorial of negative(s) not defined !');

end

end

4. Write MATLAB **function [in, fr] = infr(x)** that takes an array **x** of real numbers and returns arrays **in** and **fr** holding the integral and fractional parts, respectively, of all numbers in the array **x**.

Ans 4 :

function [in, fr] = infr(x)

in = floor(x);

fr = x - in;

end

5. Given an array **b** and a positive integer **m** create an array **d** whose entries are those in the array **b** each replicated m-times. Write MATLAB **function d = repel(b, m)** that generates array **d** as described in this problem.

Ans 5 :

function d = repel(b, m)

d = [];

for i = 1:length(b),

for j = 1:m,

d(end + 1) = b(i);

end

end

end

6. In this exercise you are to write MATLAB **function d = rep(b, m)** that has more functionality than the function **repel** of Problem 5. It takes an array of numbers **b** and the array **m** of positive integers and returns an array **d** whose each entry is taken from the array **b** and is duplicated according to the corresponding value in the array **m**. For instance, if

**b = [ 1 2]** and **m = [2 3]**, then **d = [1 1 2 2 2]**.

Ans 6 :

function d = rep(b, m)

d = [];

for i=1:length(b),

for j=1:m(i),

d(end+1) = b(i);

end

end

end

7. A *checkerboard* matrix is a square block diagonal matrix, i.e., the only nonzero entries are in the square blocks along the main diagonal. In this exercise you are to write MATLAB **function A = mysparse(n)** that takes an odd number **n** and returns a checkerboard matrix

as shown below

**A = mysparse(3)**

A =

1 0 0

0 1 2

0 3 4

**A = mysparse(5)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A = |  | | | | |
|  | 1 | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 2 | 0 | 0 |
|  | 0 | 3 | 4 | 0 | 0 |
|  | 0 | 0 | 0 | 2 | 3 |
|  | 0 | 0 | 0 | 4 | 5 |

**A = mysparse(7)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A = |  | | | | | | |
|  | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
|  | 0 | 3 | 4 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 2 | 3 | 0 | 0 |
|  | 0 | 0 | 0 | 4 | 5 | 0 | 0 |
|  | 0 | 0 | 0 | 0 | 0 | 3 | 4 |
|  | 0 | 0 | 0 | 0 | 0 | 5 | 6 |

First block in the upper-left corner is the 1-by-1 matrix while the remaining blocks are all

2-by-2.

Ans 7 :

function A = mysparse(n)

A = zeros(n);

A(1,1) = 1;

v = 2;

i = 1;

while v<n

A(v,v) = i;

A(v,v+1) = i+1;

A(v+1,v) = i+2;

A(v+1,v+1) = i+3;

v = v + 2;

i = i + 1;

end

end

8. The Legendre polynomials **Pn(x)**, **n = 0, 1, …** are defined recursively as follows

**nPn(x) = (2n-1)xPn -1 – (n-1)Pn-2(x), n = 2, 3, … , P0(x) = 1, P1(x) = x**.

Write MATLAB **function P = LegendP(n)** that takes an integer **n** – the degree of **Pn(x)** and returns its coefficient stored in the descending order of powers.

Ans 8 :

function c = LegendP(n)

c = [1 n];

for i=3:n,

c(i) = ((2\*n-1)\*i\*c(i-1) - (n-1)\*c(i-2))/n;

end

end

9. In this exercise you are to implement Euclid's Algorithm for computing the *greatest common divisor* (**gcd**) of two integer numbers **a** and **b**:

**gcd(a, 0) = a**, **gcd(a, b) = gcd(b, rem(a, b))**.

Here **rem(a, b)** stands for the remainder in dividing **a** by **b**. MATLAB has function **rem**. Write MATLAB **function gcd = mygcd(a,b)** that implements Euclid's Algorithm.

Ans 9 :

function gcd = mygcd(a,b)

if a == 0 || b == 0

gcd = a + b;

else

gcd = mygcd(b, rem(a,b));

end

end

10. The Pascale triangle holds coefficients in the series exapansion of **(1 + x)n**, where

**n = 0, 1, 2, …** . The top of this triangle, for **n = 0, 1, 2**, is shown here

**1**

**1 1**

**1 2 1**

Write MATLAB **function t = pasctri(n)** that generates the Pascal triangle **t** up to the level **n**. **Remark**. Two-dimensional arrays in MATLAB must have the same number of columns in each row. In order to aviod error messages you have to add a certain number of zero entries

to the right of last nonzero entry in each row of **t** but one. This

**t = pasctri(2)**

t =

1 0 0

1 1 0

1 2 1

is an example of the array **t** for **n = 2**.

Ans 10 :

function c = pastri(n)

n = n +1;

c = [];

%loop over matrix rows

for j=1:n,

%get elements

for i=1:j,

c(j,i) = nchoosek(j-1,i-1);

end

end

end

11. This is a continuation of Problem 10. Write MATLAB **function t = binexp(n)** that computes an array **t** with row **k+1** holding coefficients in the series expansion of **(1-x)^k**,

**k = 0, 1, ... , n**, in the ascending order of powers. You may wish to make a call from within your function to the function **pasctri** of Problem 10. Your output sholud look like this (case

**n = 3**)

**t = binexp(3)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t = |  | | | |
|  | 1 | 0 | 0 | 0 |
|  | 1 | -1 | 0 | 0 |
|  | 1 | -2 | 1 | 0 |
|  | 1 | -3 | 3 | -1 |
|  |  |  |  |  |

Ans 11 :

function t = binexp(n)

t = pastri(n);

for i=1:n+1,

for j=1:n+1,

t(i, j) = t(i, j)\*(-1)^(j+1);

end

end

end

12. MATLAB come with the built-in function **mean** for computing the *unweighted arithmetic mean* of real numbers. Let x = {x1, x2, … , xn} be an array of n real numbers. Then

In some problems that arise in mathematical statistics one has to compute the *weighted arithmetic mean* of numbers in the array x. The latter, abbreviated here as **wam**, is defined as follows

*n*

I *wk x k*

*wam*( *x*, *w*) = *k*  1

*n*

I *wk*

*k*  1

Here w = {w1, w2, … , wn} is the array of weights associated with variables x. The weights are all nonnegative with w1 + w2 + … + wn > 0.

In this exercise you are to write MATLAB **function y = wam(x, w)** that takes the arrays of

variables and weights and returns the weighted arithmetic mean as defined above. Add three error messages to terminate prematurely execution of this file in the case when:

• arrays **x** and **w** are of different lengths

• at least one number in the array w is negative

• sum of all weights is equal to zero.

Ans 12 :

function y = wam(x, w)

if length(x) ~= length(w)

error('>!> Weights amd variables of different lengths');

elseif sum(w) == 0

error('>!> Sum of all weights equal to 0');

else

for i=1:length(w),

if w(i)<0

error('>!> One of the weights less than 0 ');

end

end

end

%get weighted mean

y = sum(x.\*w)/sum(w);

end

42

13. Let w = {w1, w2, … , wn} be an array of positive numbers. The *weighted geometric mean*, abbreviated as **wgm**, of the nonnegative variables x = {x1, x2, … , xn} is defined as follows

Here we assume that the weights **w** sum up to one.

Write MATLAB **function y = wgm(x, w)** that takes arrays **x** and **w** and returns the weighted geometric mean **y** of **x** with weights stored in the array **w**. Add three error messages to

terminate prematurely execution of this file in the case when:

• arrays **x** and **w** are of different lengths

• at least one variable in the array **x** is negative

• at least one weight in the array **w** is less than or equal to zero

Also, normalize the weights **w**, if necessary, so that they will sum up to one.

Ans 13 :

function y = wgm(x,w)

if length(w) ~= length(x)

error('>!> Weights amd variables of different lengths');

elseif sum(x) ~= sum(abs(x))

error('>!> One of the variablse less than 0 ');

else

for i=1:length(w),

if w(i) <= 0

error('>!> One of the weights <= 0 ');

end

end

end

%normalise w

w = w/norm(w,1);

y = x.^w;

end

14. Write MATLAB **function [nonz, mns] = matstat(A)** that takes as the input argument a real matrix A and returns all nonzero entries of A in the column vector **nonz**. Second output parameter **mns** holds values of the unweighted arithmetic means of all columns of **A**.

Ans 14 :

function [nonz, mns] = matstat(A)

tic;

[r c] = size(A);

nonz = zeros(r\*c, 1);

mns = zeros(1, c);

%iter over rows

for i=1:r,

%iter over columns

for j=1:c,

nonz((i-1)\*c + j, 1) = A(i, j);

end

end

%get unweighted means of columns

for i=1:c,

mns(i) = mean(A(:,i));

end

toc

end

15. Solving triangles requires a bit of knowledge of trigonometry. In this exercise

you are to write MATLAB **function [a, B, C] = sas(b, A, c)** that is intended for solving triangles given two sides **b** and **c** and the angle **A** between these sides. Your function should

determine remaining two angels and the third side of the triangle to be solved. All angles should be expressed in the degree measure.

Ans 15 :

function [a, B, C] = sas(b, A, c)

a = (b^2 + c^2 - 2\*c\*b\*cosd(A))^(1/2);

C = acosd((c^2 - a^2 - b^2)/ (2\*a\*b));

B = 180-A-C;

end

16. Write MATLAB **function [A, B, C] = sss(a, b, c)** that takes three positive numbers **a**, **b**, and **c**. If they are sides of a triangle, then your function should return its angles **A**, **B**, and **C**, in the degree measure, otherwise an error message should be displayed to the screen.

Ans 16 :

function [A, B, C] = sss(a,b,c)

%determine if sides valid by

%sum of two sides properties

if a+b<c || a+c<b || c+b<a

error('>!> Sides of triangle invalid.');

end

A = acosd((a^2 - c^2 - b^2)/ (2\*b\*c));

C = acosd((c^2 - a^2 - b^2)/ (2\*a\*b));

B = 180 - A - C;

end

17. In this exercise you are to write MATLAB **function dms(x)** that takes a nonnegative number

**x** that represents an angle in the degree measure and converts it to the form

**x deg. y min. z sec.**. Display a result to the screen using commands **disp** and **sprintf**. Example:

**dms(10.2345)**

Angle = 10 deg. 14 min. 4 sec.

Ans 17 :

function dms(x)

[d, fr] = infr(x);

m = fr\*60;

s = (fr -m)\*60;

sprintf('Angle = %d deg. %d min. %d sec.', d, m, s)

end

18. Complete elliptic integral of the first kind in the Legendre form K(k2), 0 < k2 < 1,

cannot be evaluated in terms of the elementary functions. The following algorithm, due to C. F. Gauss, generates a sequence of the arithmetic means {an} and a sequence of the geometric means {bn}, where

43

It is known that both sequences have a common limit g and that an � bn, for all n. Moreover,

K(k2) =

2*g*

Write MATLAB **function K = compK(k2)** which implements this algorithm. The input parameter **k2** stands for k2. Use the loop **while** to generate consecutive members of both sequences, but do not save all numbers generated in the course of computations. Continue

execution of the **while** loop as long as an – bn � **eps**, where **eps** is the *machine epsilon*

**eps**

ans =

2.2204e-016

Add more functionality to your code by allowing the input parameter **k2** to be an array. Test your m-file and compare your results with those included here

**format long**

**compK([.1 .2 .3 .7 .8 .9])**

ans =

1.61244134872022

1.65962359861053

1.71388944817879

2.07536313529247

2.25720532682085

2.57809211334794

**format short**

Ans 18 :

function K = compK(k2)

K = zeros(1, length(k2));

for i=1:length(K),

a = 1;

b = sqrt(1 - k2(i));

while(a-b >= eps)

a = (a+b)/2;

b = sqrt(a\*b);

end

K(i) = (pi/(2\*a))/2;

end

end

19. In this exercise you are to model one of the games in the Illinois State Lottery. Three numbers, with duplicates allowed, are selected randomly from the set {0,1,2,3,4,5,6,7,8,9} in the game Pick3 and four numbers are selected in the Pick4 game. Write MATLAB **function winnumbs = lotto(n)** that takes an integer n as its input parameter and returns an array **winnumbs** consisting of **n** numbers from the set of integers described in this problem. Use MATLAB function **rand** together with other functions to generate a set of winning numbers. Add an error message that is displayed to the screen when the input parameter is out of range.

Ans 19 :

function winnumbs = lotto(n)

winnumbs = zeros(1, n);

if n>10

error('n out of range');

end

for i=1:n,

winnumbs(i) = randi([0,9]);

end

end

44

20. Write MATLAB **function t = isodd(A)** that takes an array **A** of nonzero integers and returns

**1** if all entries in the array **A** are odd numbers and **0** otherwise. You may wish to use

MATLAB function **rem** in your file.

Ans 20 :

function t = isodd(x)

t = 1; %assume all are odd

for i=1:length(x),

if rem(x(i), 2) == 0

t = 0;

break;

end

end

end

21. Given two one-dimensional arrays **a** and **b**, not necessarily of the same length. Write MATLAB **function c = interleave(a, b)** which takes arrays **a** and **b** and returns an array **c** obtained by interleaving entries in the input arrays. For instance, if **a = [1, 3, 5, 7]** and

**b = [-2, –4]**, then **c = [1, –2, 3, –4, 5, 7]**. Your program should work for empty arrays too. You cannot use loops **for** or **while**.

Ans 21 :

function c = interleave(a,b)

m = max(length(a), length(b));

if(m == length(b))

%b is bigger add elements to a

while(m - length(a) >0)

a(end + 1) = 0;

end

else

while(m - length(b) >0)

b(end + 1) = 0;

end

end

%now matrices are of equal dimensions

c = [a; b];

c = c(c~=0);

end

22. Write a script file **Problem22** to plot, in the same window, graphs of two parabolas **y = x2** and **x = y2**, where **–1** � **x** � **1**. Label the axes, add a title to your graph and use command **grid**. To improve readability of the graphs plotted add a legend. MATLAB has a command

**legend**. To learn more about this command type **help legend** in the **Command Window** and press **Enter** or **Return** key.

Ans 22 :

x = -10:10;

y1 = x.^2;

y2 = x.^0.5;

y3 = -y2;

title('my graph');

hold;

grid;

plot(x, y1);

plot(x, y2);

plot(x, y3);

xlabel('x');

ylabel('y');

23. Write MATLAB **function eqtri(a, b)** that plots the graph of the equilateral triangle with two vertices at **(a,a)** and **(b,a)**. Third vertex lies above the line segment that connects points **(a, a)** and **(b, a)**. Use function **fill** to paint the triangle using a color of your choice.

Ans 23 :

function equtri(a,b)

l = b-a;

x = a+l\*0.5;

y = a+l\*sind(60);

grid;

patch([a b x], [a a y], 'Red');

end

24. In this exercise you are to plot graphs of the Chebyshev polynomial **Tn(x)** and its first order derivative over the interval **[-1, 1]**. Write MATLAB **function plotChT(n)** that takes as the input parameter the degree **n** of the Chebyshev polynomial. Use functions **ChebT** and **derp**, included in Tutorial 2, to compute coefficients of **Tn(x)** and **T'n(x)**, respectively. Evaluate both, the polynomial and its first order derivative at **x = linspace(-1, 1)** using MATLAB function **polyval**. Add a meaningful title to your graph. In order to improve readability of your graph you may wish to add a descriptive legend. Here is a sample output

**plotChT(5)**

25. Use function **sphere** to plot the graph of a sphere of radius **r** with center at **(a, b, c)**. Use MATLAB function **axis** with an option **'equal'**. Add a title to your graph and save your computer code as the MATLAB function **sph(r, a, b, c)**.

Ans 25 :

function sph(r,a,b,c)

[x y z] = sphere;

figure;

axis equal;

surf(r\*x + a, r\*y + b, r\*z + c);

end

26. Write MATLAB **function ellipsoid(x0, y0, z0, a, b, c)** that takes coordinates (**x0, y0, z0)** of the center of the ellipsoid with semiaxes **(a, b, c)** and plots its graph. Use MATLAB functions **sphere** and **surf**. Add a meaningful title to your graph and use function **axis('equal')**.

Ans 26 :

function ellipsoid(x0, y0, z0, a, b, c)

[x y z] = sphere();

surf(x/a + x0, y/b + y0, z/c + z0);

end

27. In this exercise you are to plot a graph of the two-sided cone, with vertex at the origin, and the-axis as the axis of symmetry. Write MATLAB **function cone(a, b)**, where the input parameters **a** and **b** stand for the radius of the lower and upper base, respectively. Use MATLAB functions **cylinder** and **surf** to plot a cone in question. Add a title to your graph and use function **shading** with an argument of your choice. A sample output is shown below

Ans 27 :

function [ output\_args ] = cone( a,b )

[x y z] = cylinder([a,0,-b]);

surf(x,y,z-0.5)

end

28. The space curve r**(t) = < cos(t)sin(4t), sin(t)sin(4t), cos(4t)** >, **0** � **t** � **2**n, lies on the surface of the unit sphere **x2 + y2 + z2 = 1**. Write MATLAB script file **curvsph** that plots both the curve and the sphere in the same window. Add a meaningful title to your graph. Use MATLAB functions **colormap** and **shading** with arguments of your choice. Add the **view([150 125 50])** command.

Ans 28 :

[x y z] = sphere(50);

a = surf(x,y,z);

axis('equal')

hold on;

shading FLAT;

colormap hot;

b = ezplot3('cos(t)\*sin(4\*t)','sin(t)\*sin(4\*t)','cos(4\*t)');

set(b,'Color','Blue','LineStyle','\*')

view([150 125 50])

29. This problem requires that the professional version 5.x of MATLAB is installed.

In this exercise you are to write the m-file **secondmovie** that crates five frames of the surface

**z = sin(kx)cos(ky)**, where **0** � **x, y** � n and **k = 1, 2, 3, 4, 5**. Make a movie consisting of the

46

frames you generated in your file. Use MATLAB functions colormap and shading with

arguments of your choice. Add a title, which might look like this

Graphs of z = sin(kx)\*cos(ky), 0 <= x, y <= rt, k =1, 2, 3, 4, 5. Greek letters can be printed in the title of a graph using TeX convention, i.e., the following \pi is used to print the Greek letter 1t. Similarly, the string \alpha will be printed as u.

Ans 29 :

fig1 = figure(1);

shading FLAT;

winsize = get(fig1, 'Position');

winsize(1:2) = [0 0];

numframes = 5;

A = moviein(numframes,fig1,winsize);

set(fig1,'NextPlot','replacechildren');

t=0:pi/500:pi;

for i=1:numframes,

plot3(t,t,(sin(2\*i\*t))/2);­­­­­­

A(:,i) = getframe(fig1,winsize);

end

movie(fig1,A,30,3,winsize);

save secondmovie.mat A

Q24 Not done.