Introduction to MATLAB

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

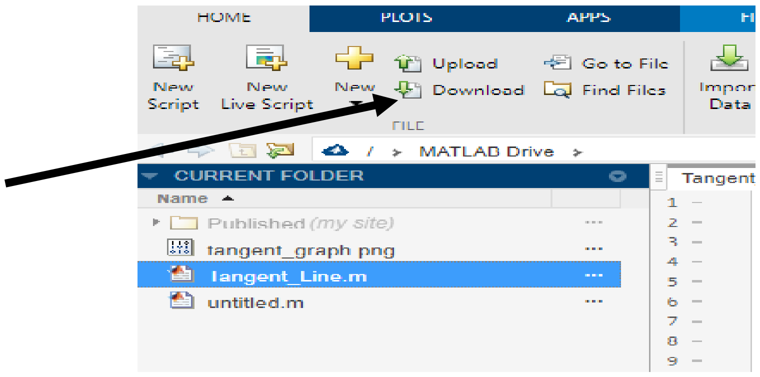
MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy-to-use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

MATLAB Online

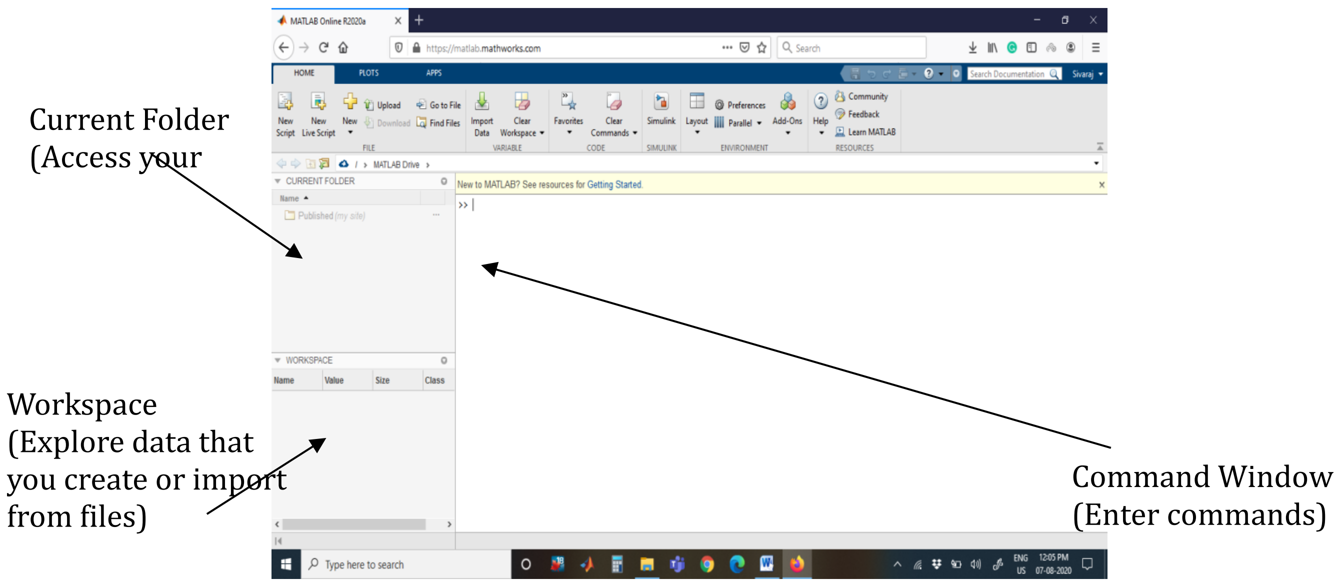
- MATLAB online can be accessed through the link https://matlab.mathworks.com/
- Enter the login credentials provided by VIT. If not sure of the password, you can reset by clicking on 'Forgot Password', after giving your VIT email ID as username.
- After entering the correct username and password, an email will be sent to your VIT email for Profile verification. Open the email and verify your profile. Now you can use the MATLAB online platform.

<u>MATLAB Drive</u>

- MATLAB Drive allows you to securely store and access your MATLAB files from anywhere. MATLAB Drive runs outside of MATLAB and is accessed from the notification area on your computer.
- The programs done on MATLAB online are stored in MATLAB drive. To copy the files to your local drive, you may download the file(s) as shown in below figure.



The graphical interface to MATLAB workspace can be seen in the following figure:



Basic MATLAB Commands

Starting with MATLAB: To start with, MATLAB can be used as a calculator. As an example of a simple interactive calculation, just type the expression you want to evaluate. Let's start at the very beginning. For example, let's suppose you want to calculate the expression, 1 + 2 × 3. You type it at the prompt command (>>) as follows:

```
>> 1+2*3
ans    =
          7
```

You would have noticed that if you do not specify an output variable, MATLAB uses a default variable `ans`, short for answer, to store the results of the current calculation. Note that the variable `ans` is created (or overwritten, if it is already existed). To avoid this, you may assign a value to a variable or output argument name. For example,

```
>> x = 1+2*3
x      =
       7
```

will result in `x` being given the value `1+2×3=7`. This variable name can always be used to refer to the results of the previous computations. Therefore, computing `4x` will result in

```
>> 4*x
ans     =
           28.0000
```

Quitting MATLAB: To end your MATLAB session, type **quit** in the Command Window, or Sign Out from your profile.

Creating MATLAB variables: MATLAB variables are created with an assignment statement. The syntax of variable assignment is
variable name = a value (or an expression)
For example,

```
>> x = expression
```

where expression is a combination of numerical values, mathematical operators, variables, and function calls. On other words, expression can involve:
• manual entry
• built-in functions

• user-defined functions

Overwriting variable: Once a variable has been created, it can be reassigned. In addition, if you do not wish to see the intermediate results, you can suppress the numerical output by putting a semicolon (;) at the end of the line. Then the sequence of commands looks like this:

```
>> t = 5;
>> t = t+1
t =
       6
```

Controlling the appearance of floating point number: MATLAB by default displays only 4 decimals in the result of the calculations. For example `x=7/6` is printed as `1.1667`, however, MATLAB does numerical calculations in double precision, which is 15 digits. The command format controls how the results of computations are displayed. Here are some examples of the different formats together with the resulting outputs.

```
>> format short
>> x=1.6667
```

If we want to see all 15 digits, we use the command format long

```
>> format long
>> x= 1.166666666666667
```

To return to the standard format, enter format short, or simply format.

Managing the workspace: The command clear or clear all removes all variables from the workspace. This frees up system memory.

```
>> clear
```

In order to display a list of the variables currently in the memory, type

```
>> who
```

while, `whos` will give more details which include size, space allocation, and class of the variables.

Getting help: To view the online documentation, select MATLAB Help from Help menu or type help directly in the command window. The preferred method is to use the Help Browser. The Help Browser can be started by selecting the **?** icon from the desktop toolbar. On the other hand, information about any command is available by typing

```
>> help <Command>
```

Vectors: We recall briefly how to enter vectors. Let a = (-1, 2, 4) and b = (1.5, 2, -1). To assign these vectors to variables a and b type either

```
>> a = [-1 2 4]
>> b = [1.5 2 -1]
```

or

```
>> a = [-1,2,4]
>> b = [1.5,2,-1]
```

Thus spaces or commas can be used.
Concatenation is the process of joining arrays to make larger ones. For example, if A=[1 2] and B=[3 4]. To create another matrix C concatenating A and B horizontally, we type

`>> C=[A B]` or `>> C=[A,B]`

Similarly vertical concatenation can be done as `>> C=[A;B]` .

One way of finding the dot product of two vectors, say a.b. Here is another which uses the important idea of element by element multiplication. Typing

```
>> c=a.*b
```

where there is a dot before the multiplication sign *, multiplies the vectors `a` and `b` element-by-element: in this case, `c = (-1.5,4, -4)`. The dot product is then obtained by summing the elements of c:

```
        >> sum(c)
```

gives `a.b=-1.5`. Similarly

```
        >> sqrt(sum(a.*a))
```

gives the magnitude of `a`. In fact, the MATLAB command, norm, will directly find the magnitude of a vector.

Matrices: A matrix is a two-dimensional array of numbers. To create a matrix in MATLAB, we enter each row as a sequence of comma or space delimited numbers, and then use semicolons to mark the end of each row. For example, consider:

$$A=\begin{bmatrix}-1 & 6\\ 7 & 11\end{bmatrix}$$

This matrix is entered in MATLAB using the following syntax:

```
        >> A = [-1,6; 7,11];
```

or consider the matrix:

$$B=\begin{bmatrix}2 & 0 & 1\\ -1 & 7 & 4\\ 3 & 0 & 1\end{bmatrix}$$

We enter it in MATLAB in the following way:

```
        >> B = [2,0,1;-1,7,4; 3,0,1]
```

Special Matrix Type: The identity matrix is a square matrix that has ones along the diagonal and zeros elsewhere. To create an $n \times n$ identity matrix, type the following MATLAB command:

```
eye(n)
```

Let’s create a 3 × 3 identity matrix:

```
        >> eye(3)
        ans =
                   1     0     0
                   0     1     0
                   0     0     1
```

To create an n × n matrix of zeros, we type zeros(n). We can also create a m × n matrix of zeros by typing zeros(m, n). Similarly, we can generate a matrix completely filled with 1’s by typing ones(n) or ones(m,n).

Referencing Matrix Elements: Individual elements and columns in a matrix can be referenced using MATLAB.

Consider the matrix:

```
        >> A = [1 2 3; 4 5 6; 7 8 9]
        A =
                1      2      3
                4      5      6
                7      8      9
```

We can pick out the element at row position m and column position n by typing A(m,n). For example:

```
        >> A(2,3)
        ans =
                  6
```

To reference all the elements in the ith column we write A(:,i). For example, we can pick out the second column of A:

```
        >> A(:,2)
        ans =
             2
             5
             8
```

To pick out the elements in the ith through jth columns we type A(:,i:j). Here we return the second and third columns:

```
>> A(:,2:3)
ans =
             2      3
             5      6
             8      9
```

We can pick out pieces or submatrices as well. Continuing with the same matrix, to pick out the elements in the second and third rows that are also in the first and second columns, we write:

```
>> A(2:3,1:2)
ans =
             4       5
             7       8
```

We can change the value of matrix elements using these references as well. Let's change the element in row 1 and column 1 to –8:

```
>> A(1,1) = -8
A =
            -8       2        3
             4       5        6
             7       8        9
```

Symbolic math computations

Symbolic Math Toolbox provides functions for solving, plotting, and manipulating symbolic math equations. The toolbox provides libraries of functions in common mathematical areas such as calculus, linear algebra, algebraic and ordinary differential equations, equation simplification, and equation manipulation. Symbolic Math Toolbox lets you analytically perform differentiation, integration, simplification, transforms, and equation solving. Your computations can be performed either analytically or using variable precision arithmetic, with the results displayed in mathematical typeset.

Key Features

- Symbolic integration, differentiation, transforms, and linear algebra.
- Algebraic and ordinary differential equation (ODE) solvers.
- Simplification and manipulation of symbolic expressions.
- Plotting of analytical functions in 2D and 3D.
- Variable-precision arithmetic.

Create Symbolic Numbers

You can create symbolic numbers by using `sym`. Symbolic numbers are exact representations, unlike floating-point numbers. For example,

```
>> sym(4/5)
   ans =
       4/5
```

The symbolic number is represented in exact rational form, while the floating-point number is a decimal approximation. Calculations on symbolic numbers are exact. For example,

```
>> sin(sym(pi))
ans =
0
```

while the numerical approximation for $\sin\pi$ can be obtained as below:

```
>> sin(pi)
ans =
   1.2246e-16
```

Create Symbolic Variables

There are two ways to create symbolic variables, `syms` and `sym`.
The `syms` syntax is a shorthand for `sym`.

To create symbolic variables `x` and `y` using `syms` and `sym` respectively.

```
>> syms x
>> y = sym('y')
```

With `syms`, you can create multiple variables in one command. Create the variables `a, b` and `c`.

```
>> syms a b c
```

To create many numbered variables, namely `a1, ..., a10` try as below:

```
>> A = sym('a', [1,10])
A =
[a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]
```

Create Symbolic Expressions

Suppose we want to study the quadratic function $f = ax^2 + bx + c$, we first create the symbolic variables `a, b, c`, and `x`:

```
>> syms a b c x
```

Then, assign the expression to f:

```
>> f = a*x^2 + b*x + c;
```

Calculus with Symbolic Math

Limits:

The following examples illustrate how to find the limit of a function $\lim_{x \to a} f(x)$.

Ex. 1. $\lim_{x \to \infty} \left(1 - \frac{2}{x}\right)^x$

Type the following code in a script file and run.

```
syms x
f(x)=(1-2/x)^x;
L = limit(f(x), x, inf)
```

Output:

```
L=
exp(-2)
```

Ex. 2. $\lim_{x \to \infty} \left(\sqrt{9x^2 + x} - 3x\right)$

Type the following code in a script file and run.

```
syms x
f = sqrt(9*x^2 + x) - 3*x
L = limit(f, x, inf)
```

Output:

```
L=
1/6
```

Ex. 3. Calculating the left and right limits of $\dfrac{x^2-1}{x^2-6x+5}$ as $x \to 5$.

Type the following code in a script file and run.

```
syms x
f = (x^2 - 1) / (x^2 - 6*x + 5)
LL = limit(f, x, 5, 'left')      % For Left limit  x→5⁻
RL = limit(f, x, 5, 'right')     % For right limit  x→5⁺
L=limit(f, x, 5)                 % For limit  x→5
```

Output:

```
LL=
-Inf

RL=
Inf

L =
NaN
```

Derivative as Rate of Change:

As we know, the derivative or slope of the tangent of $f(x)$ at a point $x=a$ can be determined by $\underset{x\to a}{Lim}\left\{\dfrac{f(x)-f(a)}{x-a}\right\}$ or $\underset{h\to 0}{Lim}\left\{\dfrac{f(a+h)-f(a)}{h}\right\}$. The following example illustrates how to find the derivative of a function $f(x)$ at a point $x=a$.

```
clear all
clc
syms h x y
f(x)=4*x-x^2;
a=1;
Df=limit((f(x+h)-f(x))/h,h,0)  % Derivative of f(x)
m=limit((f(x)-f(a))/(x-1),x,a) % slope of Tangent Line, method 1
m=limit((f(a+h)-f(1))/h,h,0)   % slopt of Tangent Line, method 2
g=f(a)+m*(x-a)
Tangent_Line=y==g
```

Output:

```
Df=

4-2*x

m=
2

m=
2

Tangent_Line =
y == 2*x + 1
```

Derivative of a $f(x)$ can be found directly by the syntax `diff(f(x),x)` and second derivative by `diff(f(x),x,2)`. The following code illustrates how to find derivative of $f(x)=\dfrac{x^2+4x+3}{\sqrt{x}}$.

```
syms x
f(x)=(x^2+4*x+3)/sqrt(x)
df_dx = diff(f(x), x)        %First derivative
d2_fx = diff(f(x), x,2)      %Second derivative
```

Output:

```
df_dx =
(2*x+4)/x^(1/2)-(x^2+4*x+3)/(2*x^(3/2))
d2_fx =
(3*(x^2+4*x+3))/(4*x^(5/2))-(2*x+4)/x^(3/2)+2/x^(1/2)
```

Exercise Questions:

1. Evaluate the following limits with MATLAB

   (a) $\lim_{x\to 0}\left(\dfrac{\frac{1}{x-1}+\frac{1}{x+1}}{x}\right)$ (b) $\lim_{x\to -1}\left(\dfrac{\sqrt{x^2+8}-3}{x+1}\right)$ (c) $\lim_{x\to -2}\left(\dfrac{x+2}{\sqrt{x^2+5}-3}\right)$ (d) $\lim_{x\to -3}\left(\dfrac{2-\sqrt{x^2-5}}{x+3}\right)$

   (e) $\lim_{x\to 0}\left(\dfrac{x-x\cos x}{\sin^2 3x}\right)$ (f) $\lim_{x\to 0}\left(\dfrac{x+x\cos x}{\sin x\cos x}\right)$

2. At time *t*, the postion of a body moving along the *s*- axis is $s=t^3-6t^2+9t$ m.
   (a) Find the body's acceleration each time the velocity is zero.
   (b) Find the body's speed each time the acceleration is zero.
   (c) Find the total distance travelled by the body from *t*=0 to *t*=2.

3. When a batericide was added to a nutrient broth in which bacteria were growing, the bacterium population continued to grow for a while, but then stopped growing an began to decline. The size of the population at time *t* (hours) was $b=10^6+10^4t-10^3t^2$. Find the growth rates at
   (a) *t*=0 hours (b) *t*=5 hours (c) *t*=10 hours

4. Find the following derivatives using MATLAB commands

   (a) $\dfrac{d}{dx}(5x^3-x^4)^7$ (b) $\dfrac{d}{dx}\left(\dfrac{1}{3x-2}\right)$ (c) $\dfrac{d}{dx}(\sin^5 x)$ (d) $\dfrac{d}{dx}\left(1-\dfrac{x}{7}\right)^{-7}$

   (e) $\dfrac{d}{dx}\left(\dfrac{x}{2}-1\right)^{-10}$ (f) $\dfrac{d}{dx}\left(\dfrac{x^2}{8}+x-\dfrac{1}{x}\right)^4$ (g) $\dfrac{d}{dx}\sqrt{3x^2-4x+6}$

5. A new product placed in market becomes very popular. Its quantity sold N is given as a function of time t, where t is measured in weeks: $N(t) = \frac{250000\,t^2}{(2t+1)^2}$, $t > 0$. Differentiate this function. Then use the derivative to evaluate $N'(52)$ and $N'(208)$ and interpret these results.
(Hints: $N'(t) = \frac{500000\,t}{(2t+1)^3}$; $N'(52) \approx 22.5$ and $N'(208) \approx 1.4$. The slopes of the tangent lines, representing the change in numbers of units sold per week, are levelling off as t increases. Perhaps the market is becoming saturated with this product: while sales continue to increase, the rate of the sales increase per week is levelling off.)

6. Suppose that an object travels so that its distance s in miles, from its starting point is a function of time t, in hours, as follows: $s(t) = 10\,t^2$, $t > 0$.
(i) Find the average velocity between the times $t = 2$ hr and $t = 5$ hr,
(ii) Find the (instantaneous) velocity when $t = 4$ hr and
(iii) acceleration.
(Hints: $Avg = \frac{s(5)-s(2)}{3} = 70\,\frac{\text{mi}}{\text{hr}}$; $s'(4) = 80\,\frac{\text{mi}}{\text{hr}}$; $20\frac{\text{mi}}{\text{hr}^2}$)

7. The United States population is becoming more diverse. Based on the U.S. Census population projections for 2000 to 2050, the projected Hispanic population (in millions) can be modelled by the function $H(t) = 37.791(1.021)^t$, where $t = 0$ corresponds to 2000 and $0 \le t \le 50$. Use H to estimate the average rate of change in the Hispanic population from 2000 to 2010.
(Hints: $\text{vg} = \frac{\text{H}(10)-\text{H}(0)}{10} = 0.873$. Based on this model, it is estimated that the Hispanic population in the United States increased, on average, at a rate of about 873,000 people per year between 2000 and 2010.)

8. A leaking oil well off the Gulf Coast is spreading a circular film of oil over the water surface. At any time t (in minutes) after the beginning of the leak, the radius of the circular oil slick (in feet) is given by $r(t) = 4t$. Find the rate of change of the area of the oil slick with respect to time.
(Hints: Rate of change in the radius over time is $\frac{\text{dr}}{\text{dt}} = 4$. The area of the oil slick is $\text{A}(\text{r}) = \pi \text{r}^2$. The rate of change of the area of the oil slick is $\frac{\text{dA}}{\text{dt}} = 32\pi\text{t}$).

9. The sales as a function of time is given by $\frac{100000}{1+100e^{-0.3\,t}}$. Find the rate of change of sales after 4 years.
(Hints: $\text{s}'(\text{t}) = \frac{3000000e^{-0.3\,t}}{(1+100e^{-0.3\,t})^2}$; $\text{s}'(4) \approx 933$. The rate of change of sales after 4 years is about 933 units per year. The positive number indicates that sales are increasing at this time).

10. Based on projections from the Kelly Blue Book, the resale value of a 2010 Toyota Corolla 4-door sedan can be approximated by the following function $f(t) = 15450 - 13915\log(t+1)$ where t is the number of years since 2010. Find and interpret $f(4)$ and $f'(4)$.
(Hints: $\text{f}(4) \approx 5500.08$. The average resale value of a 2010 Toyota Corolla in 2014 would be approximately \$5500.08. $\text{f}'(\text{t}) = -\frac{13915}{\text{t}+1}$; $\text{f}'(4) \approx -1208.64$. In 2014, the average resale value of a 2010 Toyota Corolla is decreasing by \$1208.64 per year).

Appendix

Matrices Operations		
Operation	Syntax	Remarks
Matrix Transpose	`>> C=A'`	
Matrix Inverse	`>> C=inv(A)`	In case of Singular matrix warning will be displayed.
Determinant	`>> C=det(A)`	
Addition	`>> C=A+B`	Matrices A and B must be of same order
Subtraction	`>> D=A-B`	Matrices A and B must be of same order
Multiplication	`>> C=A*B`	Matrices A and B must be compatible for multiplication (A is of order mxp and B is pxn, then C is of order mxn.)
Element-wise multiplication	`>> C=A.*B`	Eg. A=[x, y]; B=[a, b], then A.*B=[xa, yb] Matrices A and B must be of same order
Element-wise division	`>> C=A./B`	Eg. A=[x, y]; B=[a, b], then A./B=[x/a, y/b] Matrices A and B must be of same order
Element-wise exponentiation	`>> C=A.^p`	Eg. A=[x, y]; p=2; then A.^p=[x^2, y^2]
Right division	`>> A\b`	Solves the system of linear equations AX=b.
To create an array of n elements	`>>x=linspace(x1,xn,n)`	Creates an array of n elements starting with x1 and ending with xn.
To create an array with h spacing	`>> C=a:h:b`	Creates an array with first element as a and last element as b with step length h.
Determinant	`>> C=det(A)`	
Zero matrix	`>> A=zeros(m,n)`	Creates an mxn matrix of zeros.
Identity matrix	`>> I=eye(n)`	Creates an identity matrix of order n.
Ones	`>> B=ones(m,n)`	Creates a matrix of order mxn of ones.
Length of a vector	`>> n=length(X)`	Displays the number of elements in the vector X
Size of an Array	`>> [m,n]=size(A)`	Displays the number of rows and columns of the matrix.
Diagonal elements	`>> D=diag(A)`	Creates a vector with diagonal entries of a given matrix.
Diagonal matrix	`>>A=diag(x1,x2,…,xn)`	Creates a diagonal matrix with the specified diagonal elements.

Transcendental functions		
Operation	Syntax	Remarks
e^x	`>> exp(x)`	
$\log_e x$	`>> log(x)`	Base e
$\log_{10} x$	`>> log10(x)`	Base 10
sin *x*	`>> sin(x)`	Here *x* is in radians. When *x* is in degrees, we can use sind(x), cosd(x), tand(x), secd(x), cscd(x).
cos *x*	`>> cos(x)`	
tan *x*	`>> tan(x)`	
sec *x*	`>> sec(x)`	
cosec *x*	`>> csc(x)`	

Inverse trigonometric functions	`>> asin(x)`
	`>> acos(x)`
	`>> atan(x)`	

Basic Output Commands		
Operation	Syntax	Remarks
Display	`>> display('text')`	The display command writes text.
Formatting the output	`>> format short`	Scaled fixed point format with 5 digits.
	`>> format short e`	Floating point format with 5 digits.
	`>> format long`	Scaled fixed point format with 15 digits
	`>> format long e`	Floating point format with 15 digits.
Printing the output	`>>sprintf(formatSpec, A1,…,An)`	Formats the data in arrays A1,...,An according to formatSpec in column order

Some format specifiers	`>>sprintf('%f',A)`	Try with %e, %f, and %g specifiers. For example:
		`>>x=1/exp(1);`
		`>>sprintf('%0.5f',x)`