MATLAB Laboratory 09/09/10 Lecture

Lisa A. Oberbroeckling

Loyola University Maryland
loberbroeckling@loyola.edu
Starting Matlab

Go to

http://ctx.loyola.edu

and login with your Loyola name and password...
Starting Matlab

Go to

http://ctx.loyola.edu

and login with your Loyola name and password...

Find MATLAB within the ”Academic Applications” folder
Starting Matlab

Go to

http://ctx.loyola.edu

and login with your Loyola name and password...

Find MATLAB within the "Academic Applications" folder

Matlab has eight main windows:

<table>
<thead>
<tr>
<th><strong>Window</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command Window</strong></td>
<td>Main window, enter variables, runs programs</td>
</tr>
<tr>
<td><strong>Command History Window</strong></td>
<td>Logs commands entered in the Command Window</td>
</tr>
<tr>
<td><strong>Workspace Window</strong></td>
<td>Provides information about the variables that are used</td>
</tr>
<tr>
<td><strong>Current Directory Window</strong></td>
<td>Shows the files in the current directory</td>
</tr>
<tr>
<td>Figure Window</td>
<td>Contains output from graphic commands</td>
</tr>
<tr>
<td>Editor Window</td>
<td>Creates and debugs script and function files</td>
</tr>
<tr>
<td>Help Window</td>
<td>Provides help information</td>
</tr>
<tr>
<td>Launch Pad Window</td>
<td>Provides access to tools, demos, and documentation</td>
</tr>
</tbody>
</table>

** On main screen by default
G drive
G drive
My Documents (is your G-drive!)
Drives/Directories

- G drive
- My Documents (is your G-drive!)
- H drive:
  - H:\Apps\MatlabFiles\loberbro\ma302
G drive
My Documents (is your G-drive!)
H drive:

H:\Apps\MatlabFiles\loberbro\ma302

Your own drive is C$ found under ”My Computer”. ”My Computer” is also where you can find a drive associated with your flash-drive.
Drives/Directories

- **G drive**
- **My Documents (is your G-drive!)**
- **H drive:**
  - `H:\Apps\MatlabFiles\loberbro\ma302`
- **Your own drive is C$ found under ”My Computer”. ”My Computer” is also where you can find a drive associated with your flash-drive.**
- **PAY ATTENTION TO THE Current Directory!**
To type a command the cursor must be placed next to the command prompt (\texttt{>>}).
To type a command the cursor must be placed next to the command prompt (>>).

Press **Enter** for the command to be executed. Multiple commands can be typed by typing a comma (,) between them.
To type a command the cursor must be placed next to the command prompt (>>).

Press Enter for the command to be executed. Multiple commands can be typed by typing a comma (,) between them.

A semicolon (;) at the end of a command suppresses the screen output.
To type a command the cursor must be placed next to the command prompt (>>).

Press **Enter** for the command to be executed. Multiple commands can be typed by typing a comma (,) between them.

A semicolon (;) at the end of a command suppresses the screen output.

Upper and lower case characters are not equivalent; i.e., MATLAB is CASE SENSITIVE.
To type a command the cursor must be placed next to the command prompt (>>).

Press Enter for the command to be executed. Multiple commands can be typed by typing a comma (,) between them.

A semicolon (;) at the end of a command suppresses the screen output.

Upper and lower case characters are not equivalent; i.e., MATLAB is CASE SENSITIVE.

The up and down arrow keys can be used to scroll through previous commands. Also an old command can be recalled by typing the first few characters followed by the up arrow.
Good commands

- `clc` Clears the command window of all previous commands and output

- `clear` Clears all defined variables in memory. Be careful with this command! You can also clear certain variable names by typing `clear varname1 varname2`.

- `who` Lists current variables.

- `clf` Clears the current figure (plot).

- `help topic` Displays the help for `topic`. For example, `help clf`.

- `format` Sends the output display back to the default format (if you had used the command `format type` to put output into formatting `type`).

- `disp('text')` Displays `text` as output.

- `exit` or `quit` to quit MATLAB. OR use the MENU option under "File".
Good commands

- `clc` Clears the command window of all previous commands and output
- `clear` Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variable names by typing `clear varname1 varname2`
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variablenames by typing **clear varname1 varname2**
- **who** Lists current variables
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variable names by typing clear varname1 varname2
- **who** Lists current variables
- **clf** Clears the current figure (plot)
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variable names by typing `clear varname1 varname2`
- **who** Lists current variables
- **clf** Clears the current figure (plot)
- **help topic** Displays the help for `topic`. For example, `help clf`
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variable names by typing `clear varname1 varname2`
- **who** Lists current variables
- **clf** Clears the current figure (plot)
- **help topic** Displays the help for `topic`. For example, `help clf`
- **format** sends the output display back to the default format (if you had used the command `format type` to put output into formatting type)
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variable names by typing `clear varname1 varname2`
- **who** Lists current variables
- **clf** Clears the current figure (plot)
- **help topic** Displays the help for `topic`. For example, `help clf`
- **format** sends the output display back to the default format (if you had used the command `format type` to put output into formatting type)
- **disp('text')** Displays `text` as output
Good commands

- **clc** Clears the command window of all previous commands and output
- **clear** Clears all defined variables in memory CAREFUL WITH THIS COMMAND! You can also clear certain variables by typing `clear varname1 varname2`
- **who** Lists current variables
- **clf** Clears the current figure (plot)
- **help topic** Displays the help for `topic`. For example, `help clf`
- **format** sends the output display back to the default format (if you had used the command `format type` to put output into formatting type)
- **disp('text')** Displays text as output
- **exit** or **quit** to quit Matlab OR use the MENU option under ”File”
### Built-In Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sqrt(x)</code></td>
<td>Square root</td>
</tr>
<tr>
<td><code>exp(x)</code></td>
<td>Exponential ($e^x$)</td>
</tr>
<tr>
<td><code>abs(x)</code></td>
<td>Absolute value</td>
</tr>
<tr>
<td><code>log(x)</code></td>
<td>Natural logarithm</td>
</tr>
<tr>
<td><code>sin(x)</code></td>
<td>Sine of x</td>
</tr>
<tr>
<td><code>cos(x)</code></td>
<td>Cosine of x</td>
</tr>
<tr>
<td><code>tan(x)</code></td>
<td>Tangent of x</td>
</tr>
<tr>
<td><code>cot(x)</code></td>
<td>Cotangent of x</td>
</tr>
<tr>
<td><code>pi</code></td>
<td>$\pi$</td>
</tr>
</tbody>
</table>

Example:

```
>> sqrt(4)
an = 2
>> pi
ans = 3.1416
```
## Built-In Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sqrt(x)</code></td>
<td>Square root</td>
</tr>
<tr>
<td><code>exp(x)</code></td>
<td>Exponential ($e^x$)</td>
</tr>
<tr>
<td><code>abs(x)</code></td>
<td>Absolute value</td>
</tr>
<tr>
<td><code>log(x)</code></td>
<td>Natural logarithm</td>
</tr>
<tr>
<td><code>sin(x)</code></td>
<td>Sine of $x$</td>
</tr>
<tr>
<td><code>cos(x)</code></td>
<td>Cosine of $x$</td>
</tr>
<tr>
<td><code>tan(x)</code></td>
<td>Tangent of $x$</td>
</tr>
<tr>
<td><code>cot(x)</code></td>
<td>Cotangent of $x$</td>
</tr>
<tr>
<td><code>pi</code></td>
<td>$\pi$</td>
</tr>
</tbody>
</table>

```matlab
goto sqrt(4)
ans =
2

goto pi
ans =
3.1416```
Math Operations

- Uses typical symbols
- "Dot" operations: useful/necessary for use with vectors and matrices
- NUMERICALLY computes: $\cos^{-1}(\sqrt{3}/2)$
Defining Scalar Variables

Variable = Numerical value or computable expression
Defining Scalar Variables

Variable \( = \) Numerical value or computable expression

\( = \) is the **assignment operator** which assigns a value to a variable

压 enter to make the assignment

\( \text{ans} \) is the value of the last expression that is not assigned

Be careful with variable names. For example, don't name a variable help

Remember:

Use semicolon (;) to suppress screen output

Multiple commands can be typed by typing a comma (,) between them.
Defining Scalar Variables

Variable = Numerical value or computable expression

- = is the assignment operator which assigns a value to a variable
- Left-hand side can include only one variable name

Remember:
- Use semicolon (;) to suppress screen output
- Multiple commands can be typed by typing a comma (,) between them.
Defining Scalar Variables

Variable  =  Numerical value or computable expression

- = is the **assignment operator** which assigns a value to a variable
- Left-hand side can include only **one** variable name
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
Defining Scalar Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>=</th>
<th>Numerical value or computable expression</th>
</tr>
</thead>
</table>

- = is the **assignment operator** which assigns a value to a variable
- Left-hand side can include only one variable name
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
- Variables must begin with a letter
Defining Scalar Variables

Variable = Numerical value or computable expression

- = is the **assignment operator** which assigns a value to a variable
- Left-hand side can include only **one** variable name
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
- Variables must begin with a letter
- Press **Enter** to make the assignment
Defining Scalar Variables

Variable \( = \) Numerical value or computable expression

- \( = \) is the **assignment operator** which assigns a value to a variable.
- Left-hand side can include only **one** variable name.
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values.
- Variables must begin with a letter.
- Press **Enter** to make the assignment.
- **ans** is the value of the last expression that is not assigned.
Defining Scalar Variables

Variable = Numerical value or computable expression

- = is the **assignment operator** which assigns a value to a variable
- Left-hand side can include only **one** variable name
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
- Variables must begin with a letter
- Press **Enter** to make the assignment
- **ans** is the value of the last expression that is not assigned
- Be careful with variable names. For example, don’t name a variable **help**
Defining Scalar Variables

- Variable $=$ Numerical value or computable expression
  - $=$ is the **assignment operator** which assigns a value to a variable
  - Left-hand side can include only **one** variable name
  - Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
  - Variables must begin with a letter
  - Press **Enter** to make the assignment
  - **ans** is the value of the last expression that is not assigned
  - Be careful with variable names. For example, don’t name a variable **help**

Remember:
- Use semicolon (;) to suppress screen output
- Multiple commands can be typed by typing a comma (,) between them.
Defining Scalar Variables

Example: Assign the number 3 to variable \( a \) and 4 to variable \( b \). Print out \( \sqrt{a^2 + b^2} \) and assign the solution to the variable \( c \).
Defining Scalar Variables

**Example:** Assign the number 3 to variable `a` and 4 to variable `b`. Print out $\sqrt{a^2 + b^2}$ and assign the solution to the variable `c`.

```matlab
>> a=3; b=4; c = sqrt(a^2+b^2)
c =
5
```
Defining Scalar Variables

**Example:** Assign the number 3 to variable \( a \) and 4 to variable \( b \). Print out \( \sqrt{a^2 + b^2} \) and assign the solution to the variable \( c \).

```matlab
>> a=3; b=4; c = sqrt(a^2+b^2)
c = 5
```

**Example:** Verify

\[
\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}
\]

by calculating each side of the equation for \( x = \pi/5 \).
Defining Scalar Variables

Example: Assign the number 3 to variable \( a \) and 4 to variable \( b \). Print out \( \sqrt{a^2 + b^2} \) and assign the solution to the variable \( c \).

```matlab
>> a=3; b=4; c = sqrt(a^2+b^2)
c = 5
```

Example: Verify

\[
\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}
\]

by calculating each side of the equation for \( x = \pi/5 \).

```matlab
>> x = pi/5;
>> LHS = cos(x/2)^2, RHS = (tan(x)+sin(x))/(2*tan(x))
LHS =
0.9045
RHS =
0.9045
```
Arrays

Arrays
- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)
- Represents a point in $n$-dimensional space
  - Example: $(x,y)$ in 2D and $(x,y,z)$ in 3D

Row Vector
- Use space or comma between numbers
  - $x = [1 \ 2 \ 3]$

Column Vector
- Use semicolon between numbers
  - $x = [1; 2; 3]$
Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)
- Represents point in $n$-dimensional space
  - Ex: $(x, y)$ in 2D and $(x, y, z)$ in 3D
Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)
- Represents point in $n$-dimensional space
  - Ex: $(x, y)$ in 2D and $(x, y, z)$ in 3D
- **Row Vector** (Use space or comma between numbers)

```
>> x = [1 2 3]
x =
1 2 3
```
Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)
- Represents point in $n$-dimensional space
  Ex: $(x, y)$ in 2D and $(x, y, z)$ in 3D
- **Row Vector** (Use space or comma between numbers)
  ```matlab
  >> x = [1 2 3]
  x =
  1 2 3
  ```
- **Column Vector** (Use semicolon between numbers)
  ```matlab
  >> x = [1; 2; 3]
  x =
  1
  2
  3
  ```
Arrays

- **Constant Spaced Vectors:**
  - From \( m \) to \( n \), incremented by \( q \)
  
  \[
  \text{variable} = [m : q : n]
  \]

- Example:
  
  \[
  x = \text{linspace}(0,1,5)
  \]
  
  \[
  x = [0 \ 0.2500 \ 0.5000 \ 0.7500 \ 1.0000]
  \]
**Arrays**

- **Constant Spaced Vectors:**
  - From $m$ to $n$, incremented by $q$

```plaintext
variable = [m : q : n]
```

```plaintext
>> x = [1:2:7]
x =
1 3 5 7
```
**Arrays**

- **Constant Spaced Vectors:**
  - From $m$ to $n$, incremented by $q$

  \[
  \text{variable} = [m : q : n]
  \]

  ```matlab
  >> x = [1:2:7]
  x =
  1 3 5 7
  ```

  - From $m$ to $n$ with $q$ elements

  \[
  \text{variable} = \text{linspace}(m,n,q)
  \]

  `linspace` stands for linearly spaced
  If no $q$ is given, will give you 100 elements
**Constant Spaced Vectors:**

- From $m$ to $n$, incremented by $q$
  
  \[
  \text{variable} = [m : q : n]
  \]

  ```matlab
  >> x = [1:2:7]
  x =
  1 3 5 7
  ```

- From $m$ to $n$ with $q$ elements
  
  \[
  \text{variable} = \text{linspace}(m,n,q)
  \]
  
  `linspace` stands for linearly spaced
  
  If no $q$ is given, will give you 100 elements

  ```matlab
  >> x = linspace(0,1,5)
  x =
  0 0.2500 0.5000 0.7500 1.0000
  ```
Arrays

Two-Dimensional Array (Matrix)

- Can store information like a table
- Solve systems of equations such as

\[
\begin{align*}
2x + 3y + z &= 4 \\
x - 5y + 3z &= 3 \\
4x - 2y + 3z &= 2
\end{align*}
\]
Arrays

Two-Dimensional Array (Matrix)

- Can store information like a table
- Solve systems of equations such as

\[
\begin{align*}
2x + 3y + z &= 4 \\
x - 5y + 3z &= 3 \\
4x - 2y + 3z &= 2
\end{align*}
\]

variable = [1st row; 2nd row; ...; last row]
Two-Dimensional Array (Matrix)

- Can store information like a table
- Solve systems of equations such as

\[
\begin{align*}
2x + 3y + z &= 4 \\
x - 5y + 3z &= 3 \\
4x - 2y + 3z &= 2
\end{align*}
\]

variable = [1st row; 2nd row; \ldots; last row]

```
>> x = [ 2 3 1; 1 -5 3; 4 -2 3]
x =
\begin{bmatrix}
2 & 3 & 1 \\
1 & -5 & 3 \\
4 & -2 & 3
\end{bmatrix}
```
Vector:
- $\text{ve}(k)$ picks the $k$th element of $\text{ve}$
- $\text{ve}(m:n)$ picks the $m$th through $n$th elements of $\text{ve}$
Addressing Elements

- **Vector:**
  - \( ve(k) \) picks the \( k \)th element of \( ve \)
  - \( ve(m:n) \) picks the \( m \)th through \( n \)th elements of \( ve \)

\[
\begin{align*}
\text{>> } ve &= [1 \ 5 \ 2 \ 6 \ 8 \ 7] \\
ve &= \\
1 &\ 5 &\ 2 &\ 6 &\ 8 &\ 7 \\
\text{>> } ve(5) \\
ans &= \\
8 \\
\text{>> } ve(2:4) \\
ans &= \\
5 &\ 2 &\ 6 
\end{align*}
\]
Addressing Elements

- **Matrix:**
  - $A(m,n)$ picks the $(m, n)$th element of the matrix $A$
  - $A(m:n, p:q)$ picks the $(m : n) \times (p : q)$ submatrix of the matrix $A$

$$A = \begin{bmatrix} 1 & 4 & 2 & 3 \\ 3 & 6 & 9 & 2 \\ 1 & 4 & 9 & 7 \\ 2 & 5 & 1 & 8 \end{bmatrix}$$

$A(2,3)$

ans = 9

$A(2:4, 1:3)$

ans =
[3 6 9]
[1 4 9]
[2 5 1]
Addressing Elements

- **Matrix:**
  - \( A(m,n) \) picks the \((m,n)\)th element of the matrix \( A \)
  - \( A(m:n, p:q) \) picks the \((m:n) \times (p:q)\) submatrix of the matrix \( A \)

\[
A = \begin{bmatrix}
1 & 4 & 2 & 3 \\
3 & 6 & 9 & 2 \\
1 & 4 & 9 & 7 \\
2 & 5 & 1 & 8
\end{bmatrix}
\]

\[
>> A(2,3)
ans =
9
\]

\[
>> A(2:4, 1:3)
ans =
3 6 9 \\
1 4 9 \\
2 5 1
\]
Adding Elements

- Can add elements by using the variable within vector/matrix
- Must be of appropriate size
Adding Elements

- Can add elements by using the variable within vector/matrix
- Must be of appropriate size

```
>> B = [1 4 2 3; 3 6 9 2; 1 4 9 7]
B =
1 4 2 3
3 6 9 2
1 4 9 7
>> [B; 2 5 1 8]
an =
1 4 2 3
3 6 9 2
1 4 9 7
2 5 1 8
```
Deleting Elements

- Delete elements by assigning nothing to these elements

ve = [1 5 2 6 8 7]

ve = 1 5 2 6 8 7

ve(2:4) = []

ve = 1 8 7

B = [1 4 2 3; 3 6 9 2; 1 4 9 7]

B =
1 4 2 3
3 6 9 2
1 4 9 7

B(2:3,:) = []

B =
1 4 2 3
   4 9 7
Deleting Elements

- Delete elements by assigning nothing to these elements

```matlab
>> ve = [1 5 2 6 8 7]
ve =
1 5 2 6 8 7
>> ve(2:4) = []
ve =
1 8 7
>> B = [1 4 2 3; 3 6 9 2;1 4 9 7]
B =
1 4 2 3
3 6 9 2
1 4 9 7
>> B(2:3,:) = []
B =
1 4 2 3
```
Helpful Commands for Arrays

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>length(A)</code></td>
<td>Returns number of elements in the vector A</td>
</tr>
<tr>
<td><code>size(A)</code></td>
<td>Returns size of matrix A</td>
</tr>
<tr>
<td><code>reshape(A,m,n)</code></td>
<td>Rearranges A to have m rows and n columns (arranged column-wise)</td>
</tr>
</tbody>
</table>
Strings

- Is an array of characters
- Created by typing characters within single quotes
- Can include letters, digits, symbols and spaces

```matlab
a = 'Matlab is AWESOME'
a = Matlab is AWESOME
a(1) = ans = M
a(1:6) = ans = Matlab
a(1:6) = 'Dr. O.'
a = Dr. O. is AWESOME
```
Strings

- Is an array of characters
- Created by typing characters within single quotes
- Can include letters, digits, symbols and spaces

```matlab
>> a = 'Matlab is AWESOME'
a =
Matlab is AWESOME
>> a(1) =
ans =
M
>> a(1:6)
ans =
Matlab
```
Strings

- Is an array of characters
- Created by typing characters within single quotes
- Can include letters, digits, symbols and spaces

```matlab
>> a = 'Matlab is AWESOME'
a =
Matlab is AWESOME
>> a(1) =
ans =
M
>> a(1:6)
ans =
Matlab
>> a(1:6) = 'Dr. O.'
a =
Dr. O. is AWESOME
```
When you run the file, MATLAB executes each line as if you typed it into the command window
When you run the file, MATLAB executes each line as if you typed it into the command window.

Very useful; you can edit it, save it, execute it many times and "tweak" it to experiment with commands.
Script Files

- When you run the file, MATLAB executes each line as if you typed it into the command window.
- Very useful; you can edit it, save it, execute it many times and "tweak" it to experiment with commands.
- They are ASCII files with extension `.m`; also called m-files.
Script Files

- When you run the file, MATLAB executes each line as if you typed it into the command window.
- Very useful; you can edit it, save it, execute it many times and "tweak" it to experiment with commands.
- They are ASCII files with extension .m; also called m-files.
- MATLAB editor.
  - Many ways to run an m-file of name filename.m
    CURRENT DIRECTORY MUST BE RIGHT!
    1. Type filename in the command window.
    2. Type run filename in the command window.
    3. Type run('filename') in the command window.
    4. Within the editor window, press "F5".
    5. Within the editor window, on the Debug menu, chose run...
Script Files

- When you run the file, MATLAB executes each line as if you typed it into the command window.
- Very useful; you can edit it, save it, execute it many times and "tweak" it to experiment with commands.
- They are ASCII files with extension .m; also called m-files.
- MATLAB editor
  - Many ways to run an m-file of name filename.m
    - CURRENT DIRECTORY MUST BE RIGHT!
      1. Type `filename` in the command window.
      2. Type `run filename` in the command window.
      3. Type `run('filename')` in the command window.
      4. Within the editor window, press "F5".
      5. Within the editor window, on the Debug menu, chose run...
- To avoid extraneous output to the command window, put ",;" after var assignments.
When you run the file, MATLAB executes each line as if you typed it into the command window.

Very useful; you can edit it, save it, execute it many times and "tweak" it to experiment with commands.

They are ASCII files with extension .m; also called m-files.

MATLAB editor

- Many ways to run an m-file of name filename.m

  CURRENT DIRECTORY MUST BE RIGHT!

  1. Type `filename` in the command window.
  2. Type `run filename` in the command window.
  3. Type `run('filename')` in the command window.
  4. Within the editor window, press "F5".
  5. Within the editor window, on the Debug menu, chose run...

To avoid extraneous output to the command window, put ";" after var assignments.

Comment lines begin with `%"