

# Programming Exercise 1: Linear Regression

## Machine Learning

### Introduction

In this exercise, you will implement linear regression and get to see it work on data. Before starting on this programming exercise, we strongly recommend watching the video lectures and completing the review questions for the associated topics.

To get started with the exercise, you will need to download the starter code and unzip its contents to the directory where you wish to complete the exercise. If needed, use the `cd` command in Octave/MATLAB to change to this directory before starting this exercise.

You can also find instructions for installing Octave/MATLAB in the “Environment Setup Instructions” of the course website.

### Files included in this exercise

- `ex1.m` - Octave/MATLAB script that steps you through the exercise
- `ex1_multi.m` - Octave/MATLAB script for the later parts of the exercise
- `ex1data1.txt` - Dataset for linear regression with one variable
- `ex1data2.txt` - Dataset for linear regression with multiple variables
- `submit.m` - Submission script that sends your solutions to our servers
- [★] `warmUpExercise.m` - Simple example function in Octave/MATLAB
- [★] `plotData.m` - Function to display the dataset
- [★] `computeCost.m` - Function to compute the cost of linear regression
- [★] `gradientDescent.m` - Function to run gradient descent
- [†] `computeCostMulti.m` - Cost function for multiple variables
- [†] `gradientDescentMulti.m` - Gradient descent for multiple variables
- [†] `featureNormalize.m` - Function to normalize features
- [†] `normalEqn.m` - Function to compute the normal equations

★ indicates files you will need to complete

† indicates optional exercises

Throughout the exercise, you will be using the scripts `ex1.m` and `ex1_multi.m`. These scripts set up the dataset for the problems and make calls to functions that you will write. You do not need to modify either of them. You are only required to modify functions in other files, by following the instructions in this assignment.

For this programming exercise, you are only required to complete the first part of the exercise to implement linear regression with one variable. The second part of the exercise, which is optional, covers linear regression with multiple variables.

## Where to get help

The exercises in this course use Octave<sup>1</sup> or MATLAB, a high-level programming language well-suited for numerical computations. If you do not have Octave or MATLAB installed, please refer to the installation instructions in the “Environment Setup Instructions” of the course website.

At the Octave/MATLAB command line, typing `help` followed by a function name displays documentation for a built-in function. For example, `help plot` will bring up help information for plotting. Further documentation for Octave functions can be found at the [Octave documentation pages](#). MATLAB documentation can be found at the [MATLAB documentation pages](#).

We also strongly encourage using the online **Discussions** to discuss exercises with other students. However, do not look at any source code written by others or share your source code with others.

---

## 1 Simple Octave/MATLAB function

The first part of `ex1.m` gives you practice with Octave/MATLAB syntax and the homework submission process. In the file `warmUpExercise.m`, you will find the outline of an Octave/MATLAB function. Modify it to return a 5 x 5 identity matrix by filling in the following code:

```
A = eye(5);
```

---

<sup>1</sup>Octave is a free alternative to MATLAB. For the programming exercises, you are free to use either Octave or MATLAB.

When you are finished, run `ex1.m` (assuming you are in the correct directory, type “`ex1`” at the Octave/MATLAB prompt) and you should see output similar to the following:

```
ans =  
  
Diagonal Matrix  
  
   1   0   0   0   0  
   0   1   0   0   0  
   0   0   1   0   0  
   0   0   0   1   0  
   0   0   0   0   1
```

Now `ex1.m` will pause until you press any key, and then will run the code for the next part of the assignment. If you wish to quit, typing `ctrl-c` will stop the program in the middle of its run.

## 1.1 Submitting Solutions

After completing a part of the exercise, you can submit your solutions for grading by typing `submit` at the Octave/MATLAB command line. The submission script will prompt you for your login e-mail and submission token and ask you which files you want to submit. You can obtain a submission token from the web page for the assignment.

*You should now submit your solutions.*

You are allowed to submit your solutions multiple times, and we will take only the highest score into consideration.

---

## 2 Linear regression with one variable

In this part of this exercise, you will implement linear regression with one variable to predict profits for a food truck. Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities.

You would like to use this data to help you select which city to expand to next.

The file `ex1data1.txt` contains the dataset for our linear regression problem. The first column is the population of a city and the second column is the profit of a food truck in that city. A negative value for profit indicates a loss.

The `ex1.m` script has already been set up to load this data for you.

## 2.1 Plotting the Data

Before starting on any task, it is often useful to understand the data by visualizing it. For this dataset, you can use a scatter plot to visualize the data, since it has only two properties to plot (profit and population). (Many other problems that you will encounter in real life are multi-dimensional and can't be plotted on a 2-d plot.)

In `ex1.m`, the dataset is loaded from the data file into the variables  $X$  and  $y$ :

```
data = load('ex1data1.txt');           % read comma separated data
X = data(:, 1); y = data(:, 2);
m = length(y);                         % number of training examples
```

Next, the script calls the `plotData` function to create a scatter plot of the data. Your job is to complete `plotData.m` to draw the plot; modify the file and fill in the following code:

```
plot(x, y, 'rx', 'MarkerSize', 10);    % Plot the data
ylabel('Profit in $10,000s');           % Set the y-axis label
xlabel('Population of City in 10,000s'); % Set the x-axis label
```

Now, when you continue to run `ex1.m`, our end result should look like Figure 1, with the same red “x” markers and axis labels.

To learn more about the plot command, you can type `help plot` at the Octave/MATLAB command prompt or to search online for plotting documentation. (To change the markers to red “x”, we used the option ‘`rx`’ together with the plot command, i.e., `plot(...,[your options here],...,'rx');` )

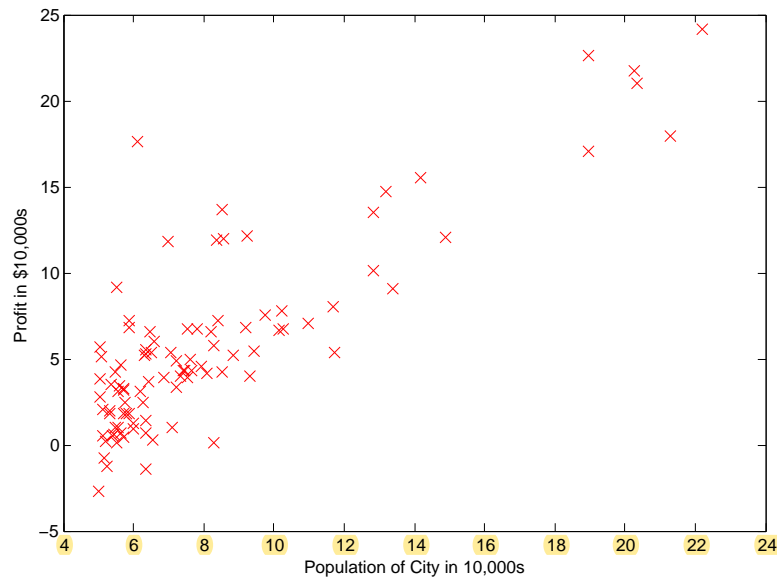


Figure 1: Scatter plot of training data

## 2.2 Gradient Descent

In this part, you will fit the linear regression parameters  $\theta$  to our dataset using gradient descent.

### 2.2.1 Update Equations


The objective of linear regression is to minimize the cost function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

where the hypothesis  $h_{\theta}(x)$  is given by the linear model

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

Recall that the parameters of your model are the  $\theta_j$  values. These are the values you will adjust to minimize cost  $J(\theta)$ . One way to do this is to use the batch gradient descent algorithm. In batch gradient descent, each iteration performs the update



$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad (\text{simultaneously update } \theta_j \text{ for all } j).$$

With each step of gradient descent, your parameters  $\theta_j$  come closer to the optimal values that will achieve the lowest cost  $J(\theta)$ .

**Implementation Note:** We store each example as a row in the the **X** matrix in Octave/MATLAB. To take into account the intercept term ( $\theta_0$ ), we add an additional first column to **X** and set it to all ones. This allows us to treat  $\theta_0$  as simply another ‘feature’.

### 2.2.2 Implementation

In `ex1.m`, we have already set up the data for linear regression. In the following lines, we add another dimension to our data to accommodate the  $\theta_0$  intercept term. We also initialize the initial parameters to 0 and the learning rate `alpha` to 0.01.

```
X = [ones(m, 1), data(:,1)]; % Add a column of ones to x
theta = zeros(2, 1); % initialize fitting parameters

iterations = 1500;
alpha = 0.01;
```

### 2.2.3 Computing the cost $J(\theta)$

As you perform gradient descent to learn minimize the cost function  $J(\theta)$ , it is helpful to monitor the convergence by computing the cost. In this section, you will implement a function to calculate  $J(\theta)$  so you can check the convergence of your gradient descent implementation.

Your next task is to complete the code in the file `computeCost.m`, which is a function that computes  $J(\theta)$ . As you are doing this, remember that the variables  $X$  and  $y$  are not scalar values, but matrices whose rows represent the examples from the training set.

Once you have completed the function, the next step in `ex1.m` will run `computeCost` once using  $\theta$  initialized to zeros, and you will see the cost printed to the screen.

You should expect to see a cost of 32.07.

*You should now submit your solutions.*

### 2.2.4 Gradient descent

Next, you will implement gradient descent in the file `gradientDescent.m`. The loop structure has been written for you, and you only need to supply the updates to  $\theta$  within each iteration.

As you program, make sure you understand what you are trying to optimize and what is being updated. Keep in mind that the cost  $J(\theta)$  is parameterized by the vector  $\theta$ , not  $X$  and  $y$ . That is, we minimize the value of  $J(\theta)$  by changing the values of the vector  $\theta$ , not by changing  $X$  or  $y$ . Refer to the equations in this handout and to the video lectures if you are uncertain.

A good way to verify that gradient descent is working correctly is to look at the value of  $J(\theta)$  and check that it is decreasing with each step. The starter code for `gradientDescent.m` calls `computeCost` on every iteration and prints the cost. Assuming you have implemented gradient descent and `computeCost` correctly, your value of  $J(\theta)$  should never increase, and should converge to a steady value by the end of the algorithm.

After you are finished, `ex1.m` will use your final parameters to plot the linear fit. The result should look something like Figure 2:

Your final values for  $\theta$  will also be used to make predictions on profits in areas of 35,000 and 70,000 people. Note the way that the following lines in `ex1.m` uses matrix multiplication, rather than explicit summation or looping, to calculate the predictions. This is an example of code vectorization in Octave/MATLAB.

*You should now submit your solutions.*

```
predict1 = [1, 3.5] * theta;  
predict2 = [1, 7] * theta;
```

## 2.3 Debugging

Here are some things to keep in mind as you implement gradient descent:

- Octave/MATLAB array indices start from one, not zero. If you're storing  $\theta_0$  and  $\theta_1$  in a vector called `theta`, the values will be `theta(1)` and `theta(2)`.
- If you are seeing many errors at runtime, inspect your matrix operations to make sure that you're adding and multiplying matrices of compatible dimensions. Printing the dimensions of variables with the `size` command will help you debug.



Figure 2: Training data with linear regression fit

- By default, Octave/MATLAB interprets math operators to be matrix operators. This is a common source of size incompatibility errors. If you don't want matrix multiplication, you need to add the “dot” notation to specify this to Octave/MATLAB. For example,  $\mathbf{A}*\mathbf{B}$  does a matrix multiply, while  $\mathbf{A}.*\mathbf{B}$  does an element-wise multiplication.

## 2.4 Visualizing $J(\theta)$

To understand the cost function  $J(\theta)$  better, you will now plot the cost over a 2-dimensional grid of  $\theta_0$  and  $\theta_1$  values. You will not need to code anything new for this part, but you should understand how the code you have written already is creating these images.

In the next step of `ex1.m`, there is code set up to calculate  $J(\theta)$  over a grid of values using the `computeCost` function that you wrote.



```

% initialize J_vals to a matrix of 0's
J_vals = zeros(length(theta0_vals), length(theta1_vals));

% Fill out J_vals
for i = 1:length(theta0_vals)
    for j = 1:length(theta1_vals)
        t = [theta0_vals(i); theta1_vals(j)];
        J_vals(i,j) = computeCost(x, y, t);
    end
end
end

```

After these lines are executed, you will have a 2-D array of  $J(\theta)$  values. The script `ex1.m` will then use these values to produce surface and contour plots of  $J(\theta)$  using the `surf` and `contour` commands. The plots should look something like Figure 3:



Figure 3: Cost function  $J(\theta)$

The purpose of these graphs is to show you that how  $J(\theta)$  varies with changes in  $\theta_0$  and  $\theta_1$ . The cost function  $J(\theta)$  is bowl-shaped and has a global minimum. (This is easier to see in the contour plot than in the 3D surface plot). This minimum is the optimal point for  $\theta_0$  and  $\theta_1$ , and each step of gradient descent moves closer to this point.

## Optional Exercises

If you have successfully completed the material above, congratulations! You now understand linear regression and should be able to start using it on your own datasets.

For the rest of this programming exercise, we have included the following optional exercises. These exercises will help you gain a deeper understanding of the material, and if you are able to do so, we encourage you to complete them as well.

---

### 3 Linear regression with multiple variables

In this part, you will implement linear regression with multiple variables to predict the prices of houses. Suppose you are selling your house and you want to know what a good market price would be. One way to do this is to first collect information on recent houses sold and make a model of housing prices.

The file `ex1data2.txt` contains a training set of housing prices in Portland, Oregon. The first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house.

The `ex1_multi.m` script has been set up to help you step through this exercise.

#### 3.1 Feature Normalization

The `ex1_multi.m` script will start by loading and displaying some values from this dataset. By looking at the values, note that house sizes are about 1000 times the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly.

Your task here is to complete the code in `featureNormalize.m` to

- Subtract the mean value of each feature from the dataset.
- After subtracting the mean, additionally scale (divide) the feature values by their respective “standard deviations.”

The standard deviation is a way of measuring how much variation there is in the range of values of a particular feature (most data points will lie within  $\pm 2$  standard deviations of the mean); this is an alternative to taking the range of values (max-min). In Octave/MATLAB, you can use the “std” function to compute the standard deviation. For example, inside `featureNormalize.m`, the quantity `X(:,1)` contains all the values of  $x_1$  (house sizes) in the training set, so `std(X(:,1))` computes the standard deviation of the house sizes. At the time that `featureNormalize.m` is called, the extra column of 1’s corresponding to  $x_0 = 1$  has not yet been added to `X` (see `ex1_multi.m` for details).

You will do this for all the features and your code should work with datasets of all sizes (any number of features / examples). Note that each column of the matrix `X` corresponds to one feature.

*You should now submit your solutions.*

**Implementation Note:** When normalizing the features, it is important to store the values used for normalization - the *mean value* and the *standard deviation* used for the computations. After learning the parameters from the model, we often want to predict the prices of houses we have not seen before. Given a new  $\mathbf{x}$  value (living room area and number of bedrooms), we must first normalize  $\mathbf{x}$  using the mean and standard deviation that we had previously computed from the training set.

## 3.2 Gradient Descent

Previously, you implemented gradient descent on a univariate regression problem. The only difference now is that there is one more feature in the matrix `X`. The hypothesis function and the batch gradient descent update rule remain unchanged.

You should complete the code in `computeCostMulti.m` and `gradientDescentMulti.m` to implement the cost function and gradient descent for linear regression with multiple variables. If your code in the previous part (single variable) already supports multiple variables, you can use it here too.

Make sure your code supports any number of features and is well-vectorized. You can use `size(X, 2)` to find out how many features are present in the dataset.

*You should now submit your solutions.*

**Implementation Note:** In the multivariate case, the cost function can also be written in the following vectorized form:

$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

where

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & \\ - & (x^{(m)})^T & - \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

The vectorized version is efficient when you're working with numerical computing tools like Octave/MATLAB. If you are an expert with matrix operations, you can prove to yourself that the two forms are equivalent.

### 3.2.1 Optional (ungraded) exercise: Selecting learning rates

In this part of the exercise, you will get to try out different learning rates for the dataset and find a learning rate that converges quickly. You can change the learning rate by modifying `ex1_multi.m` and changing the part of the code that sets the learning rate.

The next phase in `ex1_multi.m` will call your `gradientDescent.m` function and run gradient descent for about 50 iterations at the chosen learning rate. The function should also return the history of  $J(\theta)$  values in a vector `J`. After the last iteration, the `ex1_multi.m` script plots the `J` values against the number of the iterations.

If you picked a learning rate within a good range, your plot look similar Figure 4. If your graph looks very different, especially if your value of  $J(\theta)$  increases or even blows up, adjust your learning rate and try again. We recommend trying values of the learning rate  $\alpha$  on a log-scale, at multiplicative steps of about 3 times the previous value (i.e., 0.3, 0.1, 0.03, 0.01 and so on). You may also want to adjust the number of iterations you are running if that will help you see the overall trend in the curve.



Figure 4: Convergence of gradient descent with an appropriate learning rate

**Implementation Note:** If your learning rate is too large,  $J(\theta)$  can diverge and ‘blow up’, resulting in values which are too large for computer calculations. In these situations, Octave/MATLAB will tend to return NaNs. NaN stands for ‘not a number’ and is often caused by undefined operations that involve  $-\infty$  and  $+\infty$ .

**Octave/MATLAB Tip:** To compare how different learning rates affect convergence, it’s helpful to plot  $J$  for several learning rates on the same figure. In Octave/MATLAB, this can be done by performing gradient descent multiple times with a ‘hold on’ command between plots. Concretely, if you’ve tried three different values of  $\alpha$  (you should probably try more values than this) and stored the costs in  $J1$ ,  $J2$  and  $J3$ , you can use the following commands to plot them on the same figure:

```
plot(1:50, J1(1:50), 'b');
hold on;
plot(1:50, J2(1:50), 'r');
plot(1:50, J3(1:50), 'k');
```

The final arguments ‘b’, ‘r’, and ‘k’ specify different colors for the plots.

Notice the changes in the convergence curves as the learning rate changes. With a small learning rate, you should find that gradient descent takes a very long time to converge to the optimal value. Conversely, with a large learning rate, gradient descent might not converge or might even diverge!

Using the best learning rate that you found, run the `ex1_multi.m` script to run gradient descent until convergence to find the final values of  $\theta$ . Next, use this value of  $\theta$  to predict the price of a house with 1650 square feet and 3 bedrooms. You will use value later to check your implementation of the normal equations. Don't forget to normalize your features when you make this prediction!

*You do not need to submit any solutions for these optional (ungraded) exercises.*

### 3.3 Normal Equations

In the lecture videos, you learned that the closed-form solution to linear regression is

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$

Using this formula does not require any feature scaling, and you will get an exact solution in one calculation: there is no “loop until convergence” like in gradient descent.

Complete the code in `normalEqn.m` to use the formula above to calculate  $\theta$ . Remember that while you don't need to scale your features, we still need to add a column of 1's to the X matrix to have an intercept term ( $\theta_0$ ). The code in `ex1.m` will add the column of 1's to X for you.

*You should now submit your solutions.*

*Optional (ungraded) exercise:* Now, once you have found  $\theta$  using this method, use it to make a price prediction for a 1650-square-foot house with 3 bedrooms. You should find that gives the same predicted price as the value you obtained using the model fit with gradient descent (in Section 3.2.1).

## Submission and Grading

After completing various parts of the assignment, be sure to use the `submit` function system to submit your solutions to our servers. The following is a breakdown of how each part of this exercise is scored.

Part	Submitted File	Points
Warm up exercise	<code>warmUpExercise.m</code>	10 points
Compute cost for one variable	<code>computeCost.m</code>	40 points
Gradient descent for one variable	<code>gradientDescent.m</code>	50 points
Total Points		100 points

### Optional Exercises

Part	Submitted File	Points
Feature normalization	<code>featureNormalize.m</code>	0 points
Compute cost for multiple variables	<code>computeCostMulti.m</code>	0 points
Gradient descent for multiple variables	<code>gradientDescentMulti.m</code>	0 points
Normal Equations	<code>normalEqn.m</code>	0 points

You are allowed to submit your solutions multiple times, and we will take only the highest score into consideration.



# Documentation

All

Examples

Functions

## CONTENTS

## MATLAB

The Language of Technical Computing

Millions of engineers and scientists worldwide use MATLAB® to analyze and design the systems and products transforming our world. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets, and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

- ☐ [Release Notes](#)
- ☐ [PDF Documentation](#)

### Getting Started

Learn the basics of MATLAB

### Language Fundamentals

Syntax, array indexing and manipulation, data types, operators

### Data Import and Analysis

Import and export data, including large files; preprocess data, visualize and explore

### Mathematics

Linear algebra, differentiation and integrals, Fourier transforms, and other mathematics

### Graphics

Two- and three-dimensional plots, images, animation

### Programming

Scripts, functions, and classes

### App Building

App development using App Designer, GUIDE, or a programmatic workflow

### Software Development Tools

Debugging and testing, organizing large projects, source control integration, toolbox packaging

### External Language Interfaces

External language and library interfaces, including Python®, Java®, C, C++, .NET, and Web services



Environment and Settings

Preferences and settings, platform differences, adding hardware and optional features

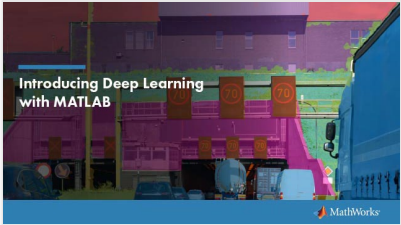
- ☐ Trial Software
- ☐ Product Updates

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

☐ Download ebook

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

☐ United States



*Join the conversation*



# Documentation

All

Examples

Functions

## CONTENTS

## Getting Started with MATLAB

The Language of Technical Computing

Millions of engineers and scientists worldwide use MATLAB® to analyze and design the systems and products transforming our world. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets, and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

## Tutorials

### Desktop Basics

Enter statements at the command line and view results.

### Matrices and Arrays

MATLAB operates primarily on arrays and matrices, both in whole and in part. A matrix is a two-dimensional array often used for linear algebra.

### Array Indexing

Variables in MATLAB are typically arrays that can hold many numbers. When you want to access selected elements of an array, use indexing.

### Workspace Variables

The workspace contains variables that you create within or import into MATLAB from data files or other programs.

### Text and Characters

Create string arrays for text, or create character arrays for data.

### Calling Functions

MATLAB provides a large number of functions that perform computational tasks. To call a function, enclose its input arguments in parentheses.

### 2-D and 3-D Plots

Graphics functions include 2-D and 3-D plotting functions to visualize data and communicate results.

### Programming and Scripts

The simplest type of MATLAB program is called a script. A script contains a sequence of commands and function calls.

### Help and Documentation

All functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax.

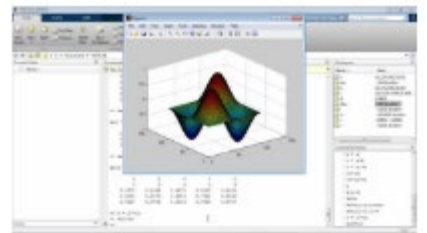
## Online Learning



### MATLAB Onramp

Free two-hour online MATLAB course

## Videos



### Getting Started with MATLAB

Get an overview of MATLAB, the language of technical computing.



### Working in the Development Environment

Access tools such as the command history workspace browser and variable editor, save and load your workspace data, and manage windows and desktop layout.

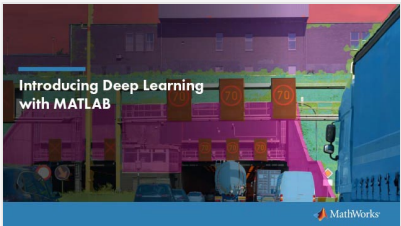
- ☐ Trial Software
- ☐ Product Updates

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

☐ Download ebook

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

☐ United States

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



*Join the conversation*

CONTENTS

## Language Fundamentals

Syntax, array indexing and manipulation, data types, operators

*MATLAB* is an abbreviation for "matrix laboratory." While other programming languages usually work with numbers one at a time, *MATLAB*® operates on whole matrices and arrays. Language fundamentals include basic operations, such as creating variables, array indexing, arithmetic, and data types.

### Entering Commands

Build and run MATLAB statements

### Matrices and Arrays

Array creation, combining, reshaping, rearranging, and indexing

### Data Types

Numeric arrays, characters and strings, tables, structures, and cell arrays; data type conversion

### Operators and Elementary Operations

Arithmetic, relational, and logical operators, special characters, rounding, set functions

### Loops and Conditional Statements

Control flow and branching using keywords, such as `if`, `for`, and `while`

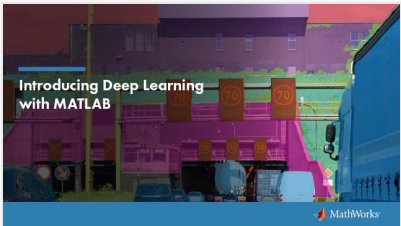
[Trial Software](#)[Product Updates](#)

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)  
© 1994-2019 The MathWorks, Inc.



Join the conversation



# Documentation

All

Examples

Functions



## CONTENTS

# Data Import and Analysis

Import and export data, including large files; preprocess data, visualize and explore

Access data from text files, spreadsheets, hardware, other software, or the web. Explore the data to identify trends, test hypotheses, and estimate uncertainty. Create customized algorithms, visualizations, and models.

### Data Import and Export

Text files, spreadsheets, and other file formats; web access

### Large Files and Big Data

Access and process collections of files and large data sets

### Preprocessing Data

Data cleaning, smoothing, grouping

### Descriptive Statistics

Range, central tendency, standard deviation, variance, correlation

### Visual Exploration

Pan, zoom, and rotate graphics; modify and save observations

[Trial Software](#)

[Product Updates](#)

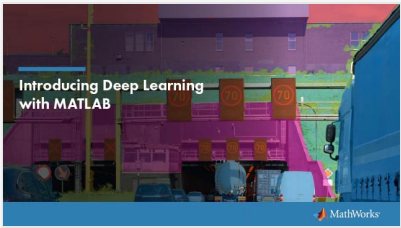


MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)  
© 1994-2019 The MathWorks, Inc.



Join the conversation



# Documentation

All

Examples

Functions



## CONTENTS

### Mathematics

Linear algebra, differentiation and integrals, Fourier transforms, and other mathematics

Math functions provide a range of numerical computation methods for analyzing data, developing algorithms, and creating models. Core functions use processor-optimized libraries for fast vector and matrix calculations.

#### Elementary Math

Trigonometry, exponentials and logarithms, complex values, rounding, remainders, discrete math

#### Linear Algebra

Linear equations, eigenvalues, singular values, decomposition, matrix operations, matrix structure

#### Random Number Generation

Seeds, distributions, algorithms

#### Interpolation

Gridded and scattered data interpolation, data gridding, piecewise polynomials

#### Optimization

Minimum of single and multivariable functions, nonnegative least-squares, roots of nonlinear functions

#### Numerical Integration and Differential Equations

Numerical integration, ordinary differential equations, delay differential equations, boundary value problems, partial differential equations

#### Fourier Analysis and Filtering

Fourier transforms, convolution, digital filtering

#### Sparse Matrices

Elementary sparse matrices, reordering algorithms, iterative methods, sparse linear algebra

#### Graph and Network Algorithms

Directed and undirected graphs, network analysis

#### Computational Geometry

Triangulation, bounding regions, Voronoi diagrams, polygons

[Trial Software](#)

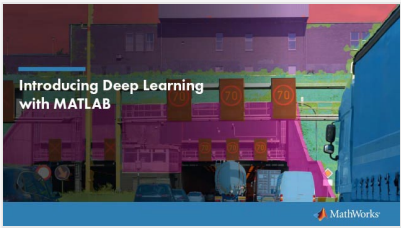
[Product Updates](#)

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



Join the conversation



# Documentation

All

Examples

Functions



## CONTENTS

## Graphics

Two- and three-dimensional plots, images, animation

Graphics functions include 2-D and 3-D plotting functions to visualize data and communicate results. Customize plots either interactively or programmatically.

### Plotting Basics

[Create 2-D Line Plot](#)

[Add Title and Axis Labels to Chart](#)

[Combine Multiple Plots](#)

[Specify Axis Limits](#)

[Create Chart with Two y-Axes](#)

### 2-D and 3-D Plots

Plot continuous, discrete, surface, and volume data

### Formatting and Annotation

Add labels, adjust colors, define axis limits, apply lighting or transparency, set camera view

### Images

Read, write, display, and modify images

### Printing and Saving

Print and export to standard file formats

### Graphics Objects

Customize graphics by setting properties of the underlying objects

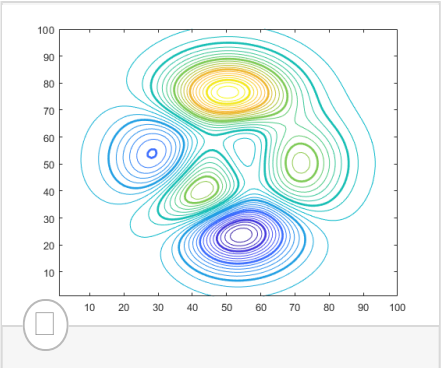
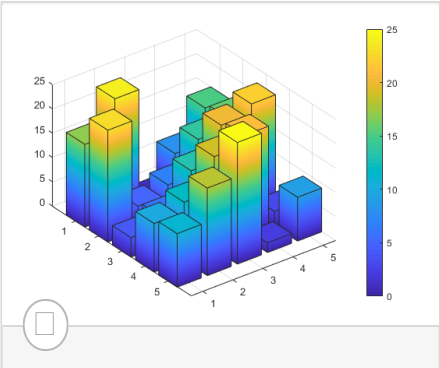
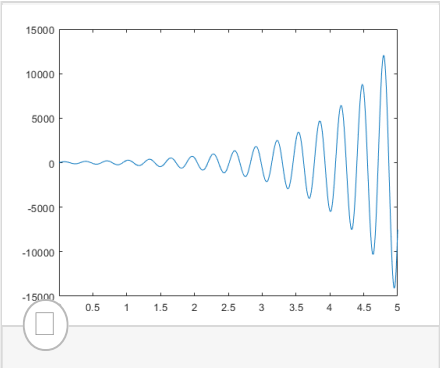
### Graphics Performance

Optimize code for improved performance

### Graphics Changes in R2014b

Migrate code from earlier releases to use the graphics system introduced in Release 2014b

## Featured Examples



Specify Axis Tick Values and Labels

Customize the tick values and labels along an axis, such as editing the tick value placement or modifying the tick label text and formatting.

Color 3-D Bars by Height

Modify a 3-D bar plot by coloring each bar according to its height.

Highlight Specific Contour Levels

Highlight contours at particular levels.

☐ Trial Software

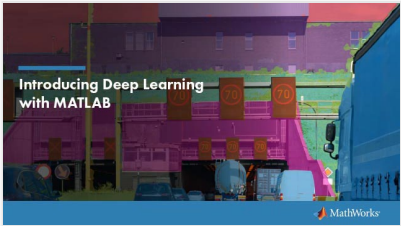
☐ Product Updates

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

☐ Download ebook

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

☐ United States

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



*Join the conversation*



# Documentation

All

Examples

Functions



## CONTENTS

## Programming

Scripts, functions, and classes

When you have a sequence of commands to perform repeatedly or that you want to save for future reference, store them in a program file. The simplest type of MATLAB® program is a [script](#), which contains a set of commands exactly as you would type them at the command line. For additional programming flexibility, create [functions](#) which accept input and return outputs. When you have specialized data structures or require many functions to interact with special kinds of data, create [classes](#) using object-oriented programming techniques.

### Scripts

Basic program files

### Functions

Programs that accept inputs and return outputs

### Live Scripts and Functions

Program files that can include formatted text, images, and output to explain the code

### Classes

Create new types of objects to use in MATLAB using object-oriented programming

### Files and Folders

File operations, MATLAB search path

### Programming Utilities

Evaluate expressions or functions indirectly, obfuscate code, set timers, handle exceptions

[Trial Software](#)

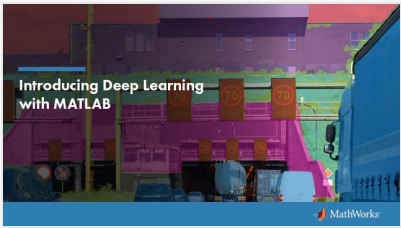
[Product Updates](#)

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

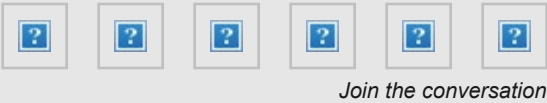
MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)







# Documentation

All

Examples

Functions



## CONTENTS

### App Building

App development using App Designer, GUIDE, or a programmatic workflow

An app is a self-contained MATLAB® program that provides a simple point-and-click interface to your code. Apps contain interactive controls such as menus, buttons, and sliders that execute specific instructions when your users interact with them. Apps can also contain plots for data visualization or interactive data exploration. Package and share your apps with other MATLAB users, or distribute them as standalone applications using MATLAB Compiler™.

There are different ways to build apps, summarized below. For a full comparison, see [Ways to Build Apps](#).

App Building Approach	Description
App Designer	This is a rich development environment that provides a large set of interactive controls, including gauges, knobs, and switches. Most graphics functionality is supported. This approach is recommended for building most apps. If you have MATLAB Compiler, you can use App Designer to create web apps.
GUIDE	This drag-and-drop environment has been available for many releases. Apps created with GUIDE are compatible with almost all other releases, and they support all the graphics functionality in MATLAB.
Programmatic Workflow	In this approach, you use MATLAB functions to create a traditional figure and place interactive components in that figure programmatically. The resulting app supports the same functionality that GUIDE apps support.

#### App Designer

Develop apps using App Designer

#### GUIDE or Programmatic Workflow

App development using GUIDE or the programmatic workflow

#### Packaging Apps

Package and share your apps

[Trial Software](#)

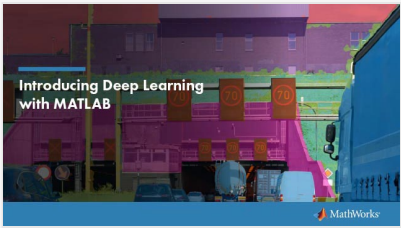
[Product Updates](#)

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



Join the conversation



# Documentation

All

Examples

Functions



## CONTENTS

## Software Development Tools

Debugging and testing, organizing large projects, source control integration, toolbox packaging

As the size and complexity of your projects grow, MATLAB® provides capabilities to support collaborative software development practices. For instance, you can integrate your MATLAB files with Git™ or Subversion® source control systems or test the functionality and performance of your code. To share code with others, package projects or other files as a toolbox.

### Debugging and Analysis

Diagnose problems, check syntax and release compatibility

### Performance and Memory

Profile code, improve performance, reduce memory requirements

### Projects

Organize large projects by managing and sharing files and settings, finding required files, and interacting with source control

### Source Control Integration

Interface MATLAB with source control system

### Testing Frameworks

Test the functionality and performance of your MATLAB code

### Toolbox Distribution

Create and share toolboxes; add documentation

☐ Trial Software

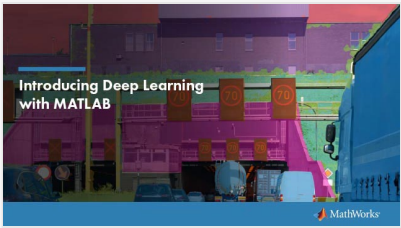
☐ Product Updates

MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



Join the conversation



# Documentation

All

Examples

Functions



## CONTENTS

# External Language Interfaces

External language and library interfaces, including Python®, Java®, C, C++, .NET, and Web services

MATLAB® provides a flexible, two-way integration with other programming languages, allowing you to reuse legacy code. To help you choose a MATLAB feature for your application, see [Integrate MATLAB with External Programming Languages and Systems](#).

## Calling Libraries in Other Languages

### C++ Libraries

Directly call C++ library functionality from MATLAB

### C Libraries

Directly call C library functions from MATLAB

### MEX File Functions

Call C/C++ or Fortran MEX file functions from MATLAB

### Java Libraries

Access Java libraries from MATLAB

### Python Libraries

Access Python functionality from MATLAB

### .NET Libraries

Access .NET libraries from MATLAB

### COM Objects

Access COM components and ActiveX® controls from MATLAB

## Calling Web Services

### HTTP Interface

Communicate with Web service from MATLAB using HTTP (Hypertext Transfer Protocol)

### WSDL (Web Services Description Language)

Communicate with Web service from MATLAB using WSDL (Web Services Description Language)

## Calling MATLAB from Other Languages

### Choosing a MATLAB API for Your Application

Determine which MATLAB API to use based on your coding environment

### Calling MATLAB from C++

Write modern C++ programs that work with MATLAB

### Calling MATLAB from Java

Write Java programs that work with MATLAB

### Calling MATLAB from Python

Write Python programs that work with MATLAB

### Calling MATLAB from C

Write C programs using mxArray that work with MATLAB

Calling MATLAB from Fortran

Write Fortran subroutines that work with MATLAB

Calling MATLAB as COM Automation Server

Write COM applications to work with MATLAB

[Trial Software](#)

[Product Updates](#)

MATLAB Documentation

[Examples](#)

[Functions](#)

[Release Notes](#)

[PDF Documentation](#)

Support

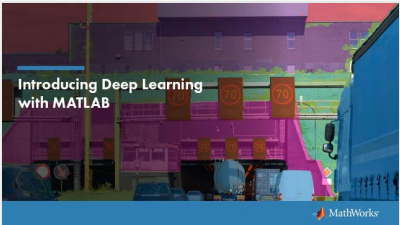
[MATLAB Answers](#)

[Installation Help](#)

[Bug Reports](#)

[Product Requirements](#)

[Software Downloads](#)



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

☐ United States

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



*Join the conversation*



# Documentation

All

Examples

Functions



## CONTENTS

## Environment and Settings

Preferences and settings, platform differences, adding hardware and optional features

The MATLAB® desktop environment helps you run commands, manage files, and view results. You can change the desktop layout and set preferences, such as fonts, keyboard shortcuts, and initial working folder.

### Startup and Shutdown

Startup command line flags, startup and shutdown files

### Basic Settings

Desktop appearance, fonts, colors, keyboard shortcuts

### Add-Ons

Find, run, and install add-ons, including optional features, apps, toolboxes, and support packages

### Platform and License

Information about current computer, license, product version

### System Commands

Interact programmatically with the operating system and the MATLAB application

### Internationalization

Locale settings and messages

### Help and Support

Product help, technical support

☐ Trial Software

☐ Product Updates

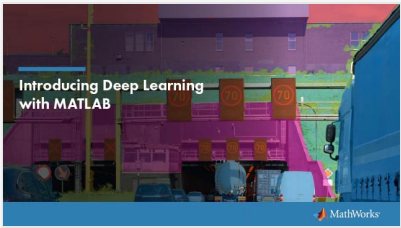


MATLAB Documentation

- Examples
- Functions
- Release Notes
- PDF Documentation

Support

- MATLAB Answers
- Installation Help
- Bug Reports
- Product Requirements
- Software Downloads



Introducing Deep Learning with MATLAB

[Download ebook](#)

Explore Products

- MATLAB
- Simulink
- Student Software
- Hardware
- Support
- File Exchange

Try or Buy

- Downloads
- Trial Software
- Contact Sales
- Pricing and Licensing
- How to Buy

Learn to Use

- Documentation
- Tutorials
- Examples
- Videos and Webinars
- Training

Get Support

- Installation Help
- Answers
- Consulting
- License Center

About MathWorks

- Careers
- Newsroom
- Social Mission
- Contact Us
- About MathWorks

MathWorks

Accelerating the pace of engineering and science

MathWorks is the leading developer of mathematical computing software for engineers and scientists.

Discover...

[United States](#)

[Patents](#) | [Trademarks](#) | [Privacy Policy](#) | [Preventing Piracy](#) | [Application Status](#)

© 1994-2019 The MathWorks, Inc.



Join the conversation