

Intro Learning Module - Find Reactions

Author: Rajesh Bhaskaran, Cornell University

[Problem Specification](#)

1. Find Reactions R_A , R_B

2. Calculate x for $r_i = 1$ cm

3. Plot x vs. r_i

4. x vs. r_i (Take 2)

5. x vs. r_i (Take 3: File Input/Output)

6. x vs. r_i (Take 4: Functions)

[Tips](#)

[Comments](#)

Find Reactions R_A , R_B

Derive Equations for R_A , R_B

$$\sum F_y = 0: R_A + R_B - 400 = 0$$

$$\sum M = 0: -3(200) - 9(200) + 12R_B = 0$$

Recast into matrix form:

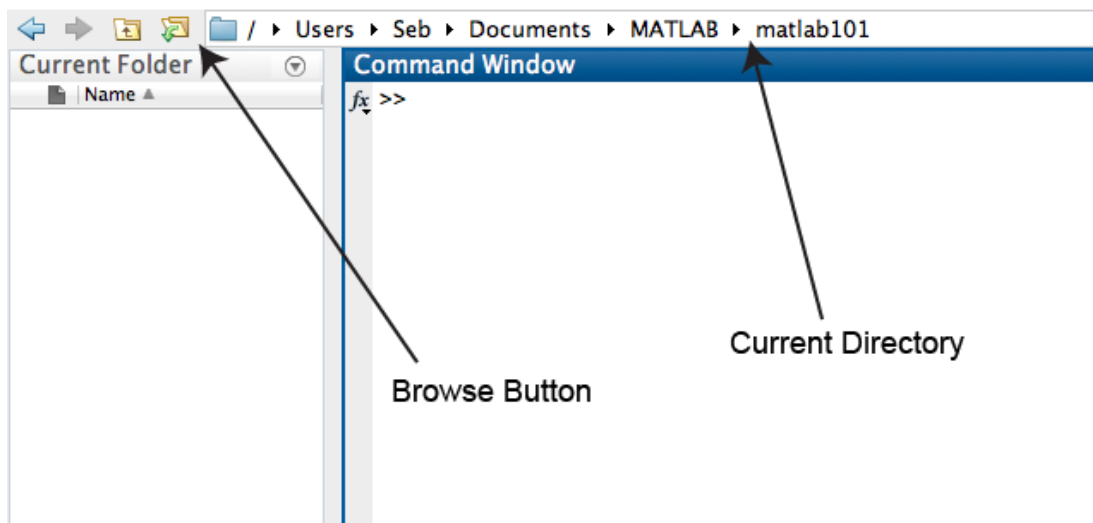
$$\begin{bmatrix} 1 & 1 \\ 0 & 12 \end{bmatrix} \begin{bmatrix} R_A \\ R_B \end{bmatrix} = \begin{bmatrix} 400 \\ 2400 \end{bmatrix} \quad or \quad [A][R] = [B]$$

We can solve this system easily. What are the resulting values for R_A and R_B ? Note these values since we'll compare them to what MATLAB reports. It is, of course, big-time overkill to solve this system using MATLAB (something probably only a government contractor would do). However, it is useful for learning the ropes, so let's go about being a government contractor.

Start MATLAB

Create a working folder called, say, *matlab101*, in a convenient location. Note the path to this folder so that you will be able to browse to it from *MATLAB*.

Launch MATLAB on your computer. Set the *matlab101* folder as your **Current Directory** by browsing to it using the browse button near the top of the MATLAB window (see image below). Confirm that the correct path appears in the **Current Directory** field.



Solve Matrix Equation in MATLAB

MATLAB comes with extensive on-line help. Let's see if this documentation can show us how to solve our simultaneous equations. From the home tab select **Help > Documentation** or simply click the **documentation icon** shown by an interrogation point. Enter the search phrase `solve simultaneous equations` and hit *Enter*. Select the hit entitled **Systems of Linear Equations**. It's usually easiest to start by looking at the examples provided which tend to be towards the bottom of the help pages. Scroll down to look at the first example under **Square Systems**.

Square Systems

The most common situation involves a square coefficient matrix **A** and a single right side column vector **b**.

Nonsingular Coefficient Matrix

If the matrix **A** is nonsingular, the solution, $x = A \backslash b$, is then the same size as **b**. For example:

```
A = pascal(3);
u = [3; 1; 4];
x = A \ u

x =
    10
   -12
     5
```

It can be confirmed that $A * x$ is exactly equal to **u**.

This example shows you how to use the `\` (mldivide) operator to solve a matrix system. This documentation page has some juicy information; glance through it for future reference.

We'll work in the **Command Window** and enter the MATLAB commands at the prompt `>>`. Create matrix **A** by entering `A = [1 1; 0 12]` at the command prompt:

```
>> A = [1 1; 0 12]
A =
     1     1
     0    12
```

A semi-colon is used to start a new matrix row. Note that MATLAB shows you the resulting values for the elements of *A*. Similarly, create matrix *B*:

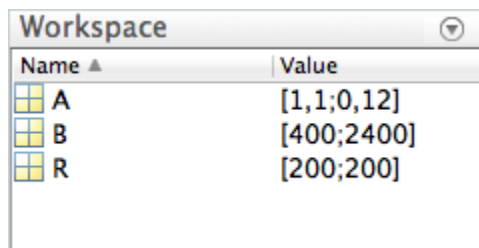
```
>> B = [400; 2400]
B =
    400
   2400
```

Calculate the reaction matrix *R* using the \ operator by entering $R = A \backslash B$.

```
>> R = A \ B
R =
    200
    200
```

Are these the values you expect?

The **Workspace** window shows all the currently defined variables. If you don't see this window, select **Home (tab) > Layout > Workspace** so that a tick mark appears next to the window name.



Name	Value
A	[1,1;0,12]
B	[400;2400]
R	[200;200]

Double-click on any variable name in the *Workspace* to take a closer peek at it.

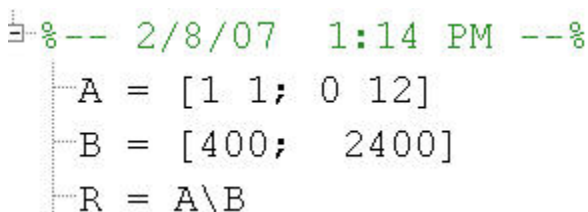
We want to start the next part of the tutorial with a clean slate. Some MATLAB commands that can help you do this are:

```
clear all; % Clear all variables from the workspace
clc; % Clear command window
```

Another way to do this is to click on the **Clear Workspace** and **Clear Commands** buttons.

Create a MATLAB Program

We have essentially used three MATLAB statements to calculate *R*. These three statements should be shown in the *Command History* window.




```
%-- 2/8/07 1:14 PM --%
A = [1 1; 0 12]
B = [400; 2400]
R = A \ B
```

If you don't see this window, select **Home (tab) > Layout > Command History** .

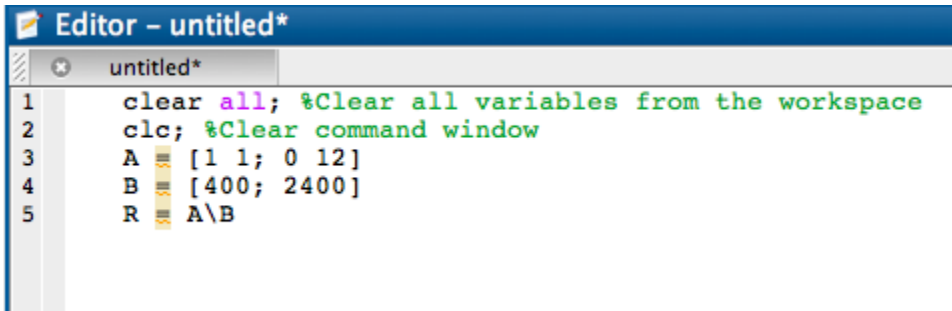
We'll create a MATLAB program to calculate R using the above three statements. Nothing fancy here: the program will just be a text file with the above statements. Such a file is called a *Script File* and have the filename extension `.m` . Let's take a peek at the documentation on *Script files*. Go in Help > Documentation and search for *Script*.

Click on the first hit and glance through this section for future reference. Also search for *Working with files and folders*.

Let's now create a Script file and simultaneously bring up the *Editor*. Select **Home (tab) > New Script**.

First, note that you can dock and undock the editor by clicking on the Show Editor Actions icon, , located in the top right corner of the Editor window. Notice that the editor tabs merges with the three default tabs when the Editor window is docked.

Now enter the above three statements in the editor (you can be lazy like me and copy-and-paste from the *Command History*; hold down the *Ctrl*-key to select multiple statements simultaneously). In addition add the clear all and clc commands at the top to start with a clean slate.



```
1 clear all; %Clear all variables from the workspace
2 clc; %Clear command window
3 A = [1 1; 0 12]
4 B = [400; 2400]
5 R = A\B
```

Save this file: Click on the **Save** icon

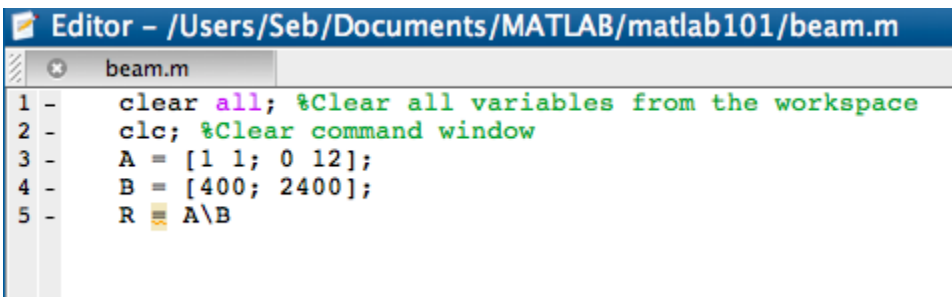
Enter beam for File name. This will create a `.m` (script) file. Make sure you are saving into your working directory. Click **Save**.

Run this program from the command line by typing in the filename *without* the `.m` suffix:

```
>> beam
```

Note that your current directory should be set correctly for this to work.

Let's say we don't want the values of A and B reported each time we run this program. To suppress the reporting of A and B values, add a semi-colon at the end of the statements for creating A and B .



```
1 - clear all; %Clear all variables from the workspace
2 - clc; %Clear command window
3 - A = [1 1; 0 12];
4 - B = [400; 2400];
5 - R = A\B
```



Save and run the program again, this time using the *Run icon* from the Editor tab. Check the result in the *Command* window. Add comments to your program to lay out, in a human language, what it does. Comments are prefixed with a `%`; See below.

```

1      % Program to solve toy beam
2      % bending problem
3      % Author: Matlab Geek
4      % Date: Dec 21, 2012
5
6      %Clear everything
7 -    clear all; %Clear all variables from the workspace
8 -    clc; %Clear command window
9
10     %Calculate reactions
11 -    A = [1 1; 0 12];
12 -    B = [400; 2400];
13 -    R = A\B

```

Any line beginning with a % will be skipped over by MATLAB; these are for human eyes only. You should add comments to your program to remind yourself what it does. Otherwise, a few months (or days!) down the line, you'll look at it, wonder what all the fuss is about and curse the ^&^* author.

Tip: To comment multiples lines at once, highlight the text then either click on the comment button, right-click and select comment or press Ctrl-R. You can uncomment in a similar manner, with the shortcut being Ctrl-T this time.

I hear you saying "this is a piece of cake, gimme more!". So let's move on to [Step 2](#).

[Go to Step 2: Calculate \$x\$ for \$r_i = 1\$ cm](#)

[Go to all MATLAB Learning Modules](#)