THE UNIVERSITY OF RHODE ISLAND

EGR 106
Foundations of Engineering II

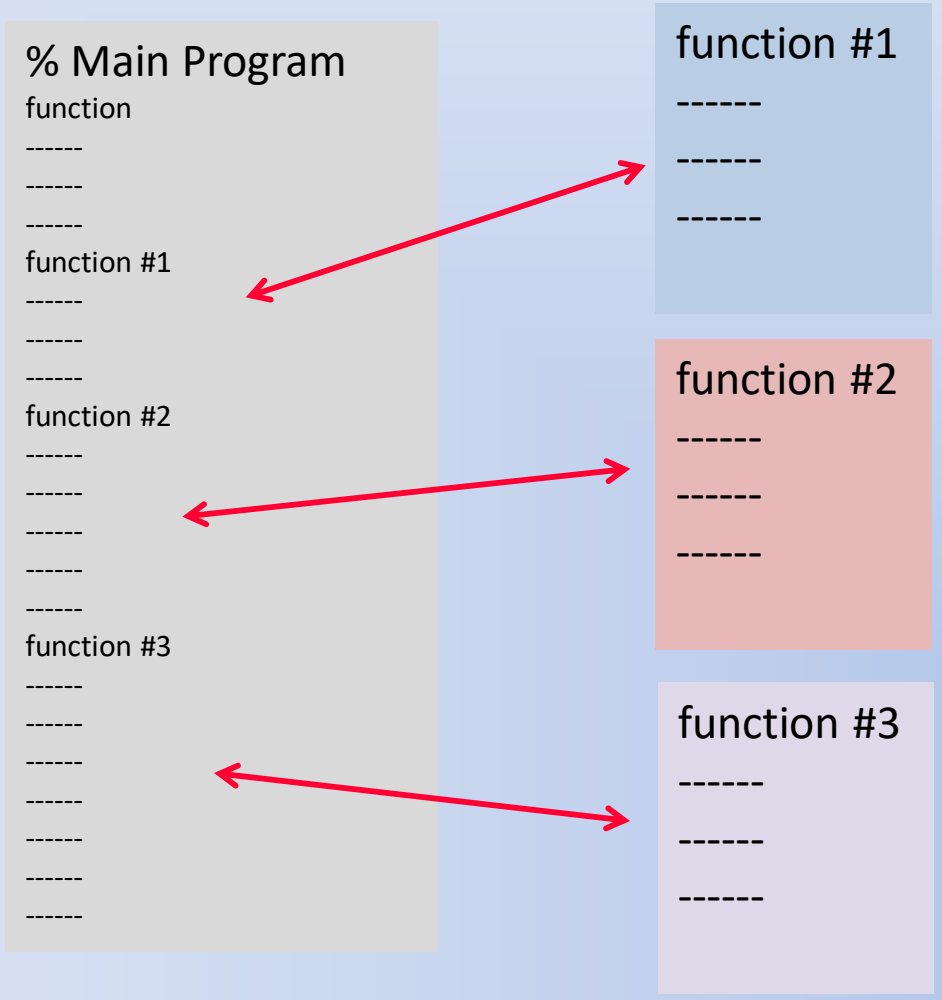
Lecture 10 – Part B
Design Project - Week 1

THINK BIG WE DO℠

This Week's Topics

- Design Project – Week 1
  - More on functions
    - Variables: Local vs. Global
  - 2D shapes
    - function 'box2d'
    - Example 1 – box_demo_1
    - Example 2 – box_demo_2
  - 3D shapes
    - function 'box'
    - Example 1 – box_demo_3
    - Example 2 – eight_color_demo
  - Viewing tools
    - function 'preview'
    - function 'model_gen'
    - function 'model_animate'

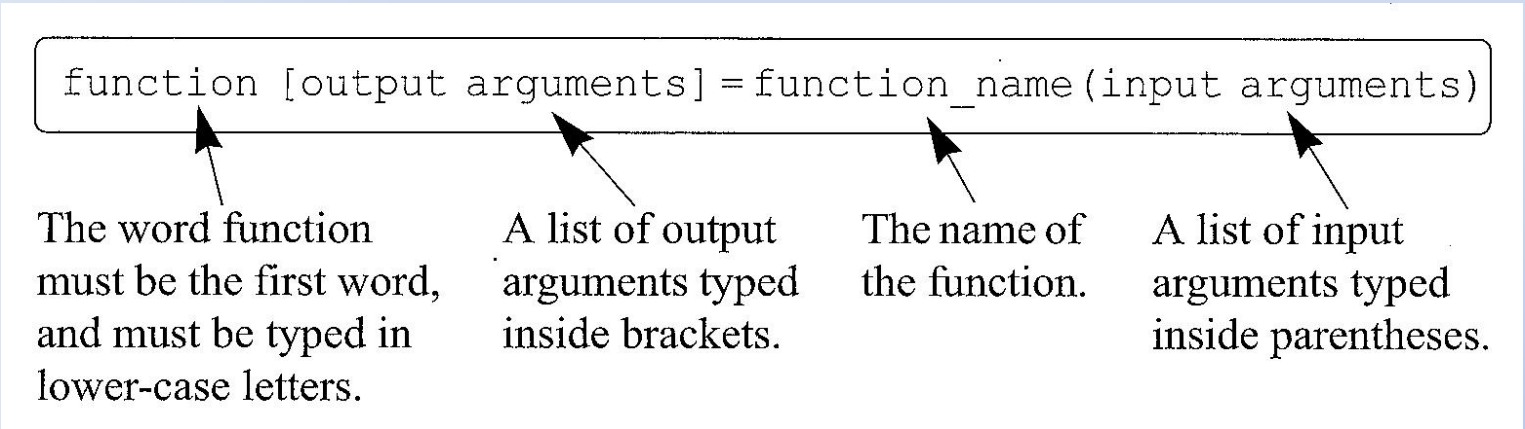
Recall Concept of a Function



Functions can be in separate files with filename: "function_name.m" or can be in the same file as main program

Function File Format

First line of the file must be of the form:



If no input or output arguments are needed:

function function_name

Input to Functions

Used to transfer data into the function from the workspace

- Workspace variables are unavailable within the function
- Any necessary variables must be brought in

For multiple inputs:

- Separate them by commas
- Order is important

Examples of built-in functions with inputs:

sum(x) **plot(x,y)**

Output from Functions

Used to transfer results back into the workspace from the function

For multiple outputs:

- Separate them by commas in brackets
- Order is important

Output variables must be assigned

Examples of built-in functions with outputs:

```
y = sum(x)
[value,location] = max(x)
```

Variables: Local vs. Global

- Functions create their own workspace with their own local variables distinct from those in the original workspace
- Functions cannot modify variables within the original workspace – except through outputs
- Exception: variables can be declared global and will then span both workspaces and be manipulated in both

```
global variable_name
```

- Hence, use of global variables provides a way to share variables between the main program workspace and the function workspace

Variables: Local vs. Global (cont.)

Example 1:

```
function main_demo_program
%
a=1; b=3; c=6; d=9;
%
[x y z]=global_demo(a,b,c);
disp(['Result: {x y z] = ',num2str([x,y,z])])

function [x y z]=global_demo(a,b,c)
% GLOBAL_DEMO demonstrates
x=a+b+c; y=a*b*c; z=c/(a*b);
```

Command Window

```
>>
Result: {x y z] = 10   18    2
>>
```

a, b and c are defined in input list

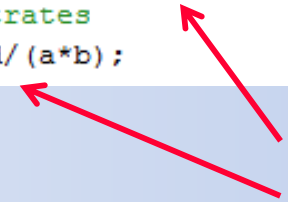
Variables: Local vs. Global (cont.)

Example 2 :

```
function main_demo_program
%
a=1; b=3; c=6; d=9;
%
[x y z]=global_demo(a,b,c);
disp(['Result: {x y z] = ',num2str([x,y,z])])

function [x y z]=global_demo(a,b,c)
% GLOBAL_DEMO demonstrates
x=a+b+c; y=a*b*c; z=d/(a*b);
```

Command Window

```
??? Undefined function or variable 'd'.

Error in ==> main_demo_program>global_demo at 13
x=a+b+c; y=a*b*c; z=d/(a*b);

Error in ==> main_demo_program at 7
[x y z]=global_demo(a,b,c);
```



d is undefined in function

Variables: Local vs. Global (cont.)

Example

```
function main_demo_program
%
global d
a=1; b=3; c=6; d=9;
%
[x y z]=global_demo(a,b,c);
disp(['Result: {x y z] = ',num2str([x,y,z])])

function [x y z]=global_demo(a,b,c)
% GLOBAL_DEMO demonstrates
%
global d
%
x=a+b+c; y=a*b*c; z=d/(a*b);
```

Command Window

```
>>
Result: {x y z] = 10   18    3
>>
```

- Each function has its own workspace
- global command allows for values to span both workspaces
- Design project functions will use a few global variables

Defining Shapes – a 2D example

Function:

box2D.m (available on Brightspace)

```
function box2D(xmin,xmax,ymin,ymax,D,C)
% creates rectangular box where xmin<=x<=xmax, ymin<=y<=ymax,
%    with D=true (solid), D=false (hole)
```

xmin, xmax - minimum and maximum x-values

ymin, ymax - minimum and maximum y-values

D = 1 or true (add points) or D=0 or false (remove points)

C – character string defining color (ex: ‘r’, ‘g’, etc.)

User defined function – box2D.m

```
function box2D(xmin,xmax,ymin,ymax,D,C)
% creates rectangular box where xmin<=x<=xmax, ymin<=y<=ymax,
%    with D=true (solid), D=false (hole)
%
global Nx Ny d color
%
for i=1:Ny
    for j=1:Nx
        x=j;
        y=i;
        if x>=xmin & x<=xmax & y>=ymin & y<=ymax
            d(i,j)=D;
            color(i,j)=C;
        end
    end
end
```

Parameters to initialize: Nx, Ny, d and color (cont.)

The main program must first:

- Clear command window, workspace variables and close open figure windows (clc, clear all, close all)
- Define Nx, Ny, d and color to be global (shared with functions)
- Nx – number of columns
- Ny – number of rows
- d – Ny x Nx logical array initially false
- color – Ny x Nx array of characters initially all spaces - char(0)

Example – Draw a 3 x 13 Blue Box (box_demo_1)

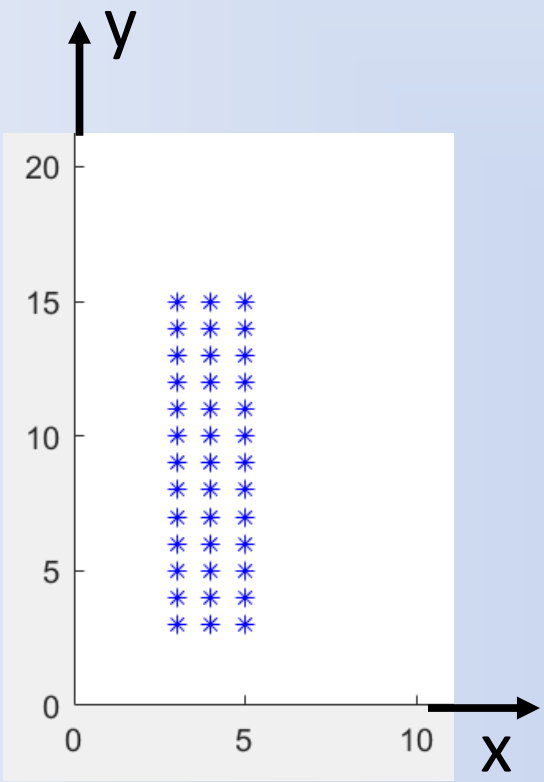
```
1  function box_demo_1
2  %
3  clc; clear all; close all
4  %
5  global Nx Ny d color
6  %
7  % define design domain (use Nx=10, Ny=20)
8  %
9  Nx=10;
10 Ny=20;
11 %
12 % initialize arrays 'd' and 'color' (Ny rows x Nx columns)
13 %
14 d=false(Ny,Nx);
15 color=char(zeros(Ny,Nx));
16 %
17 % create blue rectangular box with 3<=x<=5, 3<=y<=15
18 box2D(3,5,3,15,true,'b')
```

call to function box2D

```
19 %
20 % display result
21 %
22 for i=1:Ny
23     for j=1:Nx
24         x=j;
25         y=i;
26         if d(i,j)==true
27             marker=[color(i,j) '*'];
28             plot(x,y,marker)
29             hold on
30         end
31     end
32 end
33 axis([0 30 0 30])
34 d
35 color
```

Nested loop through each point
If d(i,j) is true, plot the point

Result - 3 x 13 Blue Box




```
d =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     1     1     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

x

y

```
color =

     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
     b     b     b
```

x

y

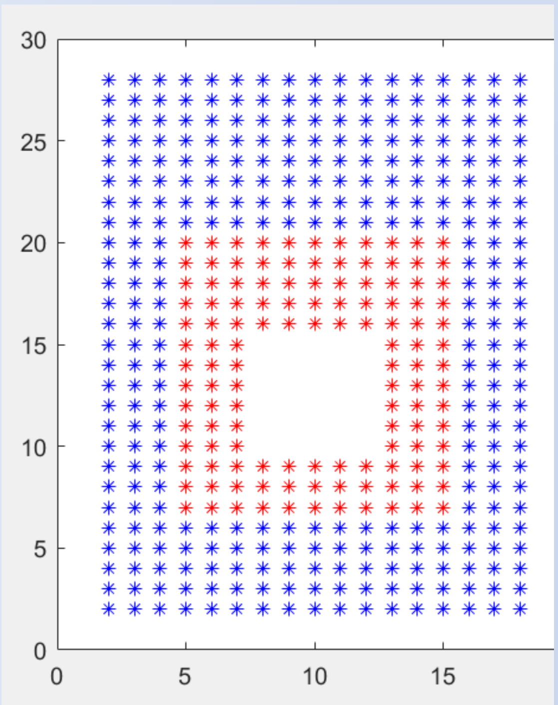
Number of rows = Ny
Number of columns = Nx
=> size(d) = Ny x Nx

Multiple Rectangles and Rectangular Holes (box_demo_2)

```
function box_demo_2
%
clc; clear all; close all
%
global Nx Ny d color
%
% define design domain (use Nx=20, Ny=30)
%
Nx=20;
Ny=30;
%
% initialize arrays 'd' and 'color' (Ny rows x Nx columns)
%
d=false(Ny,Nx);
color=char(zeros(Ny,Nx));
%
% create blue rectangular box with 2<=x<=18, 2<=y<=28
box2D(2,18,2,28,true,'b')
% create red rectangular box with 5<=x<=15, 7<=y<=20
box2D(5,15,7,20,true,'r')
% create rectangular hole with 8<=x<=12, 10<=y<=15
box2D(8,12,10,15,false,'r')
```

```
%
% display result
%
for i=1:Ny
    for j=1:Nx
        x=j;
        y=i;
        if d(i,j)==true
            marker=[color(i,j) '*'];
            plot(x,y,marker)
            hold on
        end
    end
end
axis([0 30 0 30])
d
color
```

Result – box_demo_2




```
d =
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
color =
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbrrrrrrrrrrbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
 bbbbbbbbbbbbbbbb
```

Creating 3D Boxes

Files needed

box_demo_3.m – main program

Function files:

box.p* – function to create a 3D box

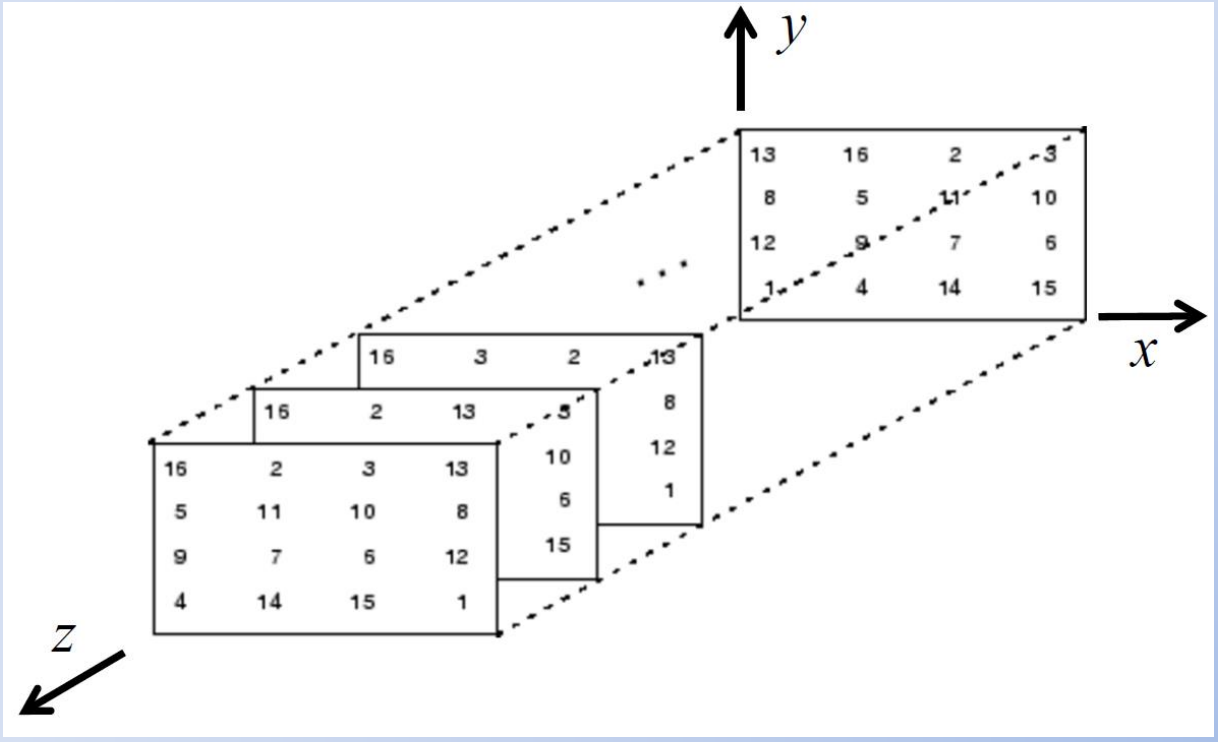
```
function box(xmin,xmax,ymin,ymax,zmin,zmax,D,C)
% creates rectangular prism for points (x,y,z), where
%  xmin<=x<=xmax, ymin<=y<=ymax, , zmin<=z<=zmax,
%    with D=1 (solid), D=0 (hole)
```

preview.p – creates a 3D figure for viewing result

*p files – protected mode Matlab function scripts (can be run but source code not viewable, cannot be edited)

Drawing 3D Shapes

3D Arrays

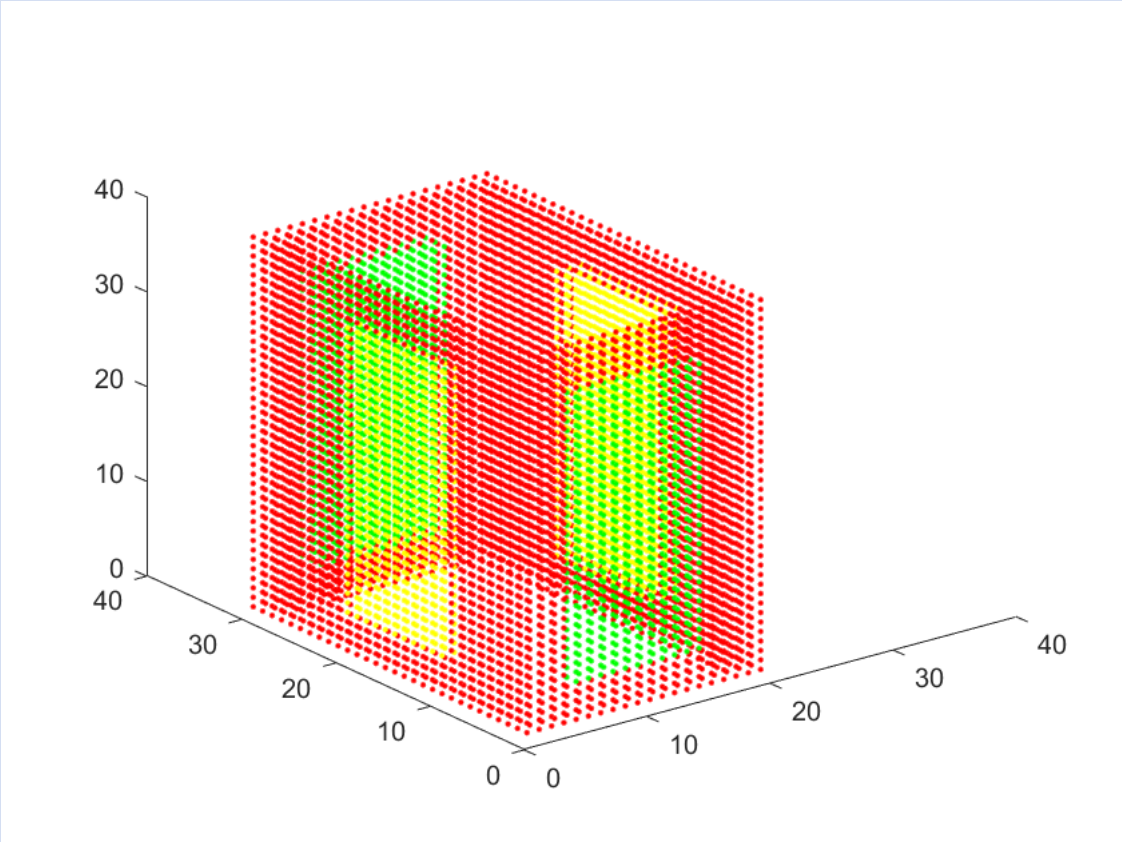


box_demo_3 (using plot3 to view)

```
clc; clear all; close all
%
global Nx Ny Nz d color
%
% Define design space
%
Nx=20;
Ny=30;
Nz=40;
%
d=false(Ny,Nx,Nz);
color=char(zeros(Ny,Nx,Nz));
%
% Build geometry
%
box(1,20,1,30,1,40,true,'r');
box(5,15,1,30,5,35,true,'g');
box(1,20,10,20,5,35,true,'y');
box(3,17,3,27,1,40,false,'y');
```

```
%
% display result
%
for i=1:Ny
    for j=1:Nx
        for k=1:Nz
            x=j;
            y=i;
            z=k;
            if d(i,j,k)==true
                marker=[color(i,j,k) '.'];
                plot3(x,y,z,marker)
                hold on
            end
        end
    end
end
axis([0 40 0 40 0 40])
```

box_demo_3 – Result with plot3



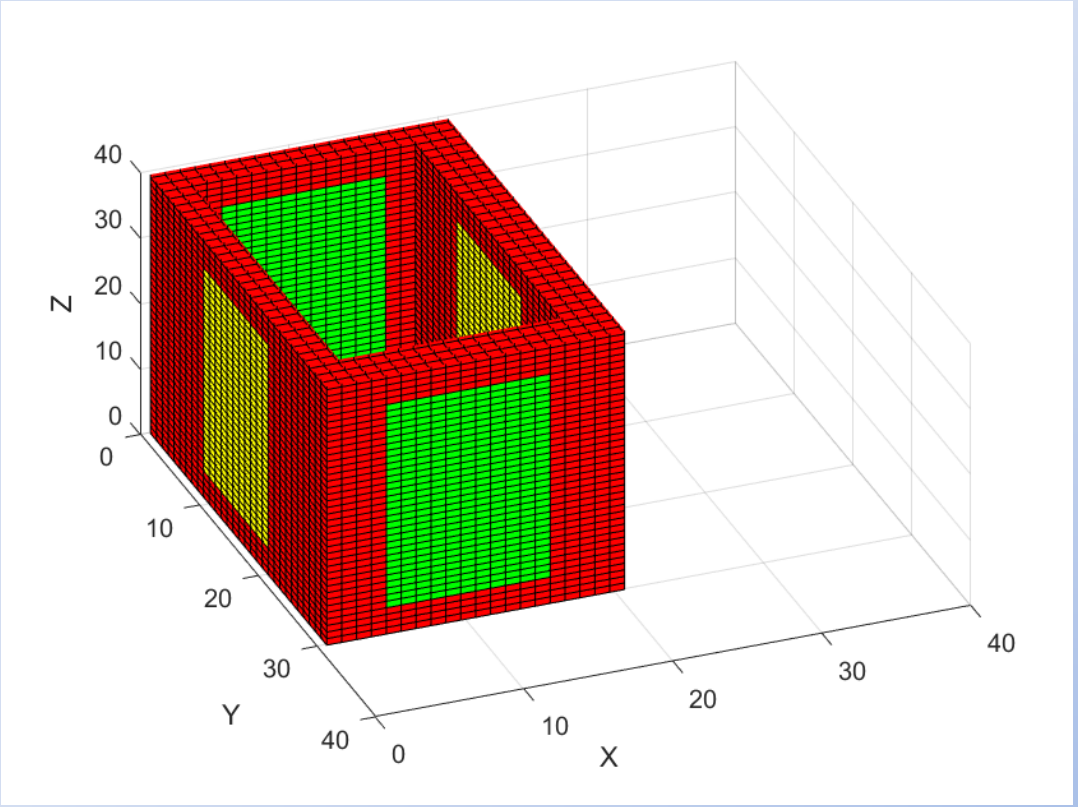
box_demo_3 (using preview to view)

```
1   function box_demo_3
2   %
3   clc; clear all; close all
4   %
5   global Nx Ny Nz d color
6   %
7   % Define design space
8   %
9   Nx=20;
10  Ny=30;
11  Nz=40;
12  %
13  d=false(Ny,Nx,Nz);
14  color=char(zeros(Ny,Nx,Nz));
```

```
15  %
16  % Build geometry
17  %
18  box(1,20,1,30,1,40,true,'r');
19  box(5,15,1,30,5,35,true,'g');
20  box(1,20,10,20,5,35,true,'y');
21  box(3,17,3,27,1,40,false,'y');
22  %
23  % Preview geometry
24  %
25  preview
```

Draws a colored unit cube at each point where d(i,j,k) = true

box_demo_3 – Result with preview



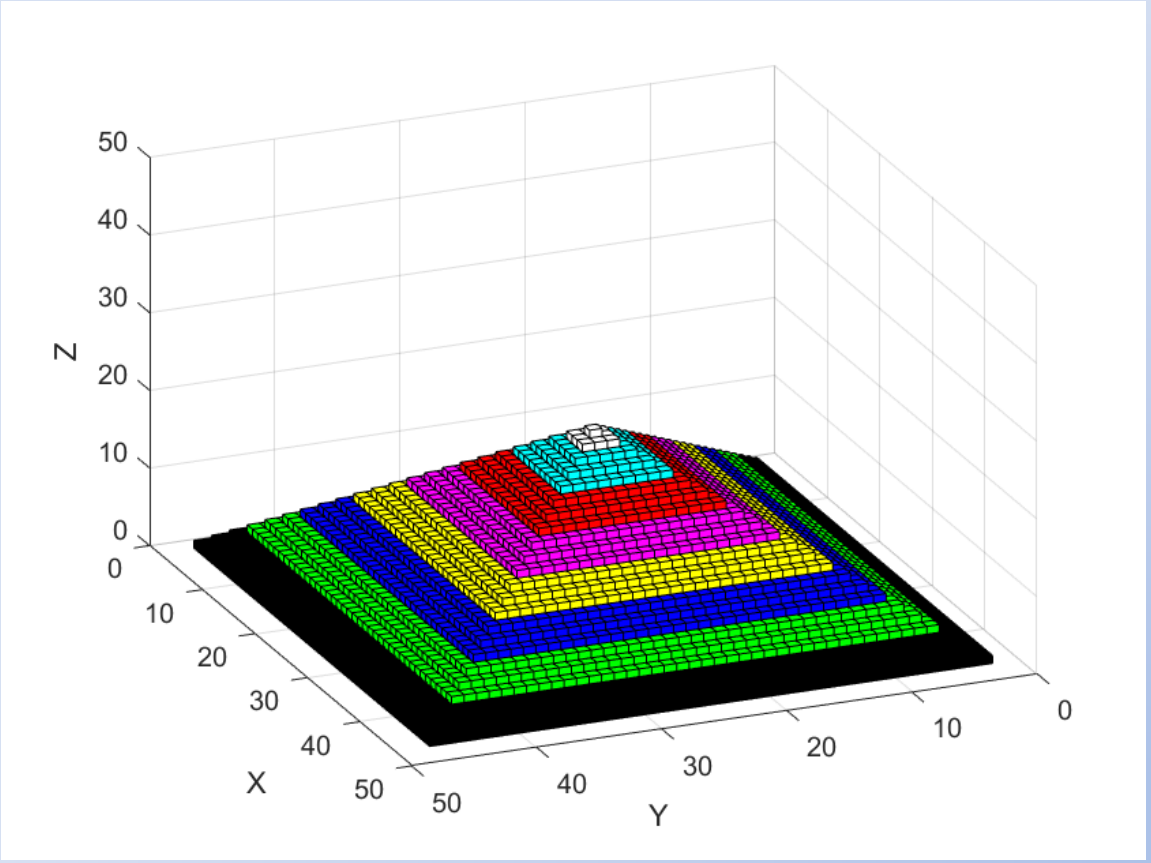
Another example: eight_color_demo.m

```
clc; clear all; close all
%
global Nx Ny Nz d color
%
% Define design space
%
Nx=50;
Ny=50;
Nz=25;
%
% Initialize array variables
%
d=false(Ny,Nx,Nz);
color=char(zeros(Ny,Nx,Nz));
```

```
%
% Build geometry
%
box( 3,47, 3,47,1,1,true,'k');
box( 4,46, 4,46,1,2,true,'k');
box( 5,45, 5,45,1,3,true,'k');
box( 6,44, 6,44,1,4,true,'g');
box( 7,43, 7,43,1,5,true,'g');
box( 8,42, 8,42,1,6,true,'g');
box( 9,41, 9,41,1,7,true,'b');
box(10,40,10,40,1,8,true,'b');
box(11,39,11,39,1,9,true,'b');
box(12,38,12,38,1,10,true,'y');
box(13,37,13,37,1,11,true,'y');
box(14,36,14,36,1,12,true,'y');
box(15,35,15,35,1,13,true,'m');
box(16,34,16,34,1,14,true,'m');
box(17,33,17,33,1,15,true,'m');
box(18,32,18,32,1,16,true,'r');
box(19,31,19,31,1,17,true,'r');
box(20,30,20,30,1,18,true,'r');
box(21,29,21,29,1,19,true,'c');
box(22,28,22,28,1,20,true,'c');
box(23,27,23,27,1,21,true,'c');
box(24,26,24,26,1,22,true,'w');
box(25,25,25,25,1,23,true,'w');
%
```

```
%
% Preview geometry
%
preview
```

eight_color_demo.m – Result with preview



More Tools for Viewing Geometry

- model_gen.p (more details next week)
  - Converts 3D geometry to CAD format (model.obj)
  - Provides smoothing of surfaces using triangular patches
  - Takes longer to run than 'preview' function
  - Can be imported to CAD viewing software (such as MeshLab)
  - Can be used for 3D printing (may need to convert to .stl in MeshLab)
- model_animate.p
  - Displays rotating view of smoothed surface
  - Saves animation as animated GIF file (model.gif)

Apply to eight_color_demo.m

```
clc; clear all; close all
%
global Nx Ny Nz d color
%
% Define design space
%
Nx=50;
Ny=50;
Nz=25;
%
% Initialize array variables
%
d=false(Ny,Nx,Nz);
color=char(zeros(Ny,Nx,Nz));
```

```
%
% Build geometry
%
box( 3,47, 3,47,1,1,true,'k');
box( 4,46, 4,46,1,2,true,'k');
box( 5,45, 5,45,1,3,true,'k');
box( 6,44, 6,44,1,4,true,'g');
box( 7,43, 7,43,1,5,true,'g');
box( 8,42, 8,42,1,6,true,'g');
box( 9,41, 9,41,1,7,true,'b');
box(10,40,10,40,1,8,true,'b');
box(11,39,11,39,1,9,true,'b');
box(12,38,12,38,1,10,true,'y');
box(13,37,13,37,1,11,true,'y');
box(14,36,14,36,1,12,true,'y');
box(15,35,15,35,1,13,true,'m');
box(16,34,16,34,1,14,true,'m');
box(17,33,17,33,1,15,true,'m');
box(18,32,18,32,1,16,true,'r');
box(19,31,19,31,1,17,true,'r');
box(20,30,20,30,1,18,true,'r');
box(21,29,21,29,1,19,true,'c');
box(22,28,22,28,1,20,true,'c');
box(23,27,23,27,1,21,true,'c');
box(24,26,24,26,1,22,true,'w');
box(25,25,25,25,1,23,true,'w');
%
```

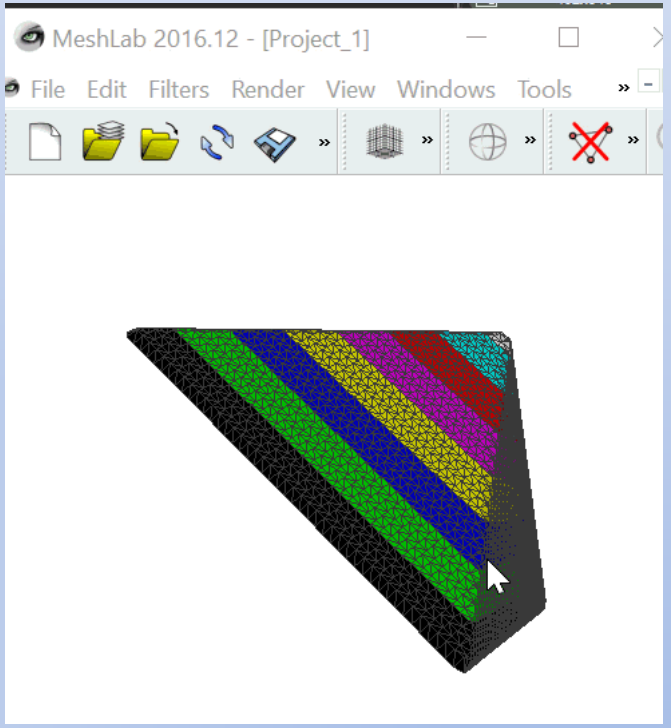
```
%
% Preview geometry
%
preview
%
% Create 3D model
%
model_gen
movefile('model.obj','eight_color_demo.obj')
%
% Create animation
%
model_animate
movefile('model.gif','eight_color_demo.gif')
```

Use movefile command to rename .obj and .gif files

eight_color_demo.m – Result with model_gen

Matlab Figure Window

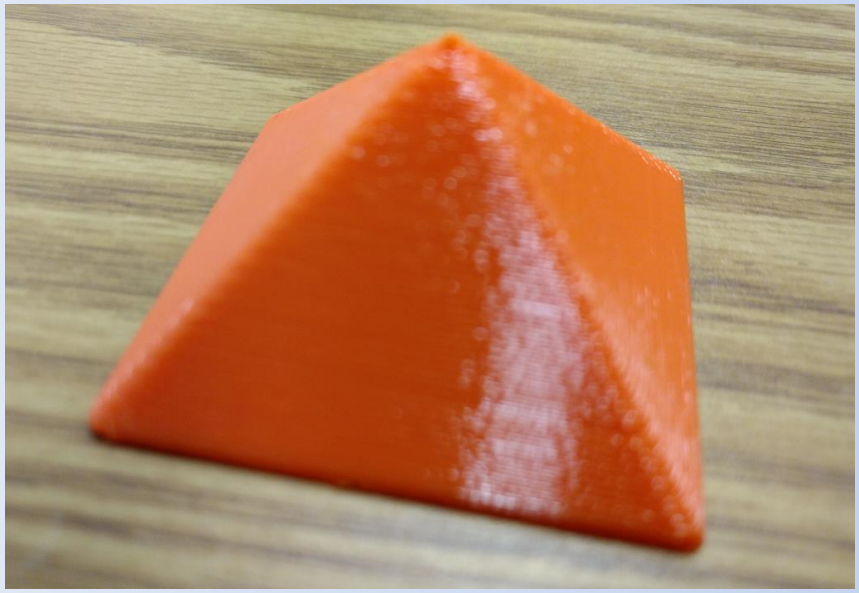
MeshLab



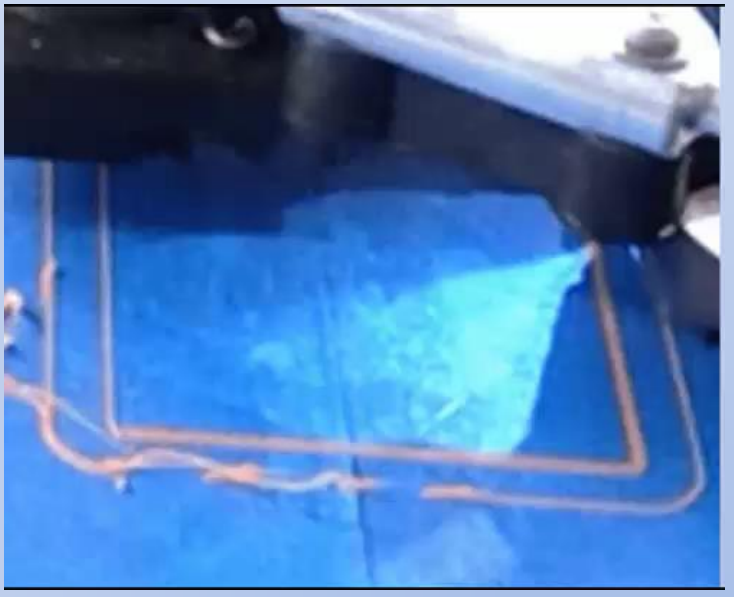
eight_color_demo.m – Result with model_animate

eight_color_demo.m – 3D print

3D Printed Part



Time Lapse Video of Printing Process



Files available on Brightspace (compressed in week_10.zip)

m files	p files*
box_demo_1.m	box.p
box_demo_2.m	preview.p
box_2d.m	model_gen.p
eight_color_demo.m	model_animate.p

*p files – protected mode Matlab function scripts (can be run but source code not viewable, cannot be edited)

Next Week

- More shape tools
  - box
  - sphere
  - cylinder_x, cylinder_y, cylinder_z
  - segment
- Creating and viewing 3D models