BURNSIM (MATLAB) INSTRUCTIONS

Graham Orr, Harvey Mudd College, FEB 2008v3

INCLUDED TOPICS:

1. BACKGROUND

Because the mass flux through a nozzle is closely correlated to surface area of the burning solid fuel grain, it may be useful to examine how certain fuel grain shapes evolve as they burn away during a motor firing. The reaction process is heterogeneous, meaning that the reaction takes place on the interface between the fuel solid and the hot gases near the fuel grain surface. As fuel is consumed from the surface, the grain shape evolves, which may have an impact on the reaction surface area and therefore thrust level.

These m-files simulate the geometric effects of a fuel grain on a rocket motor's thrust curve. In addition, it is a practice in the use of non-dimensional parameters, which is particularly useful in numerical analysis of fluid flows, computer simulations, and verification of small-scale models to larger systems.

2. OVERVIEW OF CODE

The provided code works as-is and shouldn't require modification. It would be cruel to do otherwise. Here's a brief description of each m-file:

makeBore.m - This file creates a matrix of ones and zeros that represents the cross-sectional geometry of the fuel grain of study. Zeros represent fuel space and ones represent the initial bore shape of the grain. The default resolution of this matrix is 200x200. Although in application to rocket motors the fuel burns out once the fuel burns away to the outer limit, this function just keeps all the space "outside the fuel grains as" ones. The *burnArea* file will take this matrix, apply the outer limit boundary condition and use this matrix as the initial conditions for the burn.

burnArea.m - This file actually does the burn simulation. It inputs a geometric matrix (from makeBore, although can be manually defined as discussed later) and “burns” the initial geometry outward to simulate the shapes that would occur from a burn. In this code “0” represents unburned fuel and “1” represents the gas or burned fuel region. Suppose we had an initial matrix (consisting of ones and zeros) with a discrete index denoted by *x[n]*. At *x[n+1]* the surfaces of *x[n]* would be a new loci of points some distance *d* from the surface of *x[n].* The “surface interface” is computed as:

$$\Delta x[n+1] = x[n+1] - x[n] \quad (2.1)$$

From here, we can count up the number of ones that exists in the surface interface to get an idea of the relative "surface area" that was consumed from index *n* to *n+1.* The "surface area" is reported on the right portion of the input and is non-dimensional.

Once the burn has reached the "fuel grain" boundaries, the simulation stops and a topological map of the burn is output:

$$y_{topo} \propto \sum_{k=1}^{\frac{N}{3}} \Delta x[3k] \quad (2.2)$$

This just compiles the images of each surface interface for every third index so burn propagation can easily be viewed in a printout.

3. Non-Dimensional I/O

When doing simulations or analysis between a model and a system, non-dimensional parameters can help one "scale" the model to the real system. This is because non-dimensional parameters do not depend on physical measurements but rather relative measurements or clever groupings of variables such that they have no dimension.

This code uses non-dimensional inputs as follows:

- Lengths are relative to the outer radius *R* of the fuel grain.
  - $\prod_L$=L/R (L= grainLength) (3.1)
  - $\prod_R$= R_{bore} /R (3.2)
  - i.e. if R=2.0", R_{bore}=0.7", $\prod_R$ =0.35

- Areas *A* are relative to the surface area of cylinder of radius *R*, length *L(R)*
  - $\prod_A$= A/(2πRL) (3.3)
  - i.e. if A=6 sq", R=2.0", L=6"=3R, $\prod_A$ =A/2πRL=0.0796

Similarly, the code uses non-dimensional outputs as follows:

- Areas *A* are relative to the surface area of cylinder of radius *R*, length *L(R)*
  - $A=2\pi RL \prod_A$ (3.4)
  - The amplitude on the "burn curve" is $\prod_A$

- Times *t* are relative to the time it takes for the burning-front to burn a distance *R*
  - $t=\prod_T R/r$ (*r* is linear burn rate) (3.5)
  - The time on the "burn curve" is $\prod_T$

This way you can use your geometric measurements of the fuel grains to "scale" the output curves to proper time and amplitude.

BASIC SIMULATION I//O

To do a basic run this is the structure:

$$\boldsymbol{[B\ SA]=burnArea(makeBore(\prod_{R},Q),\ \prod_{L});} \qquad (4.1)$$

Q is a choice input to determine the shape of your bore. Q is *1* for pentagram, *2* for circle, and *3* for square.

B is an *NxNxT* array, where *N* is the pixel width and *T* is the number of arrays in the trial. *B(:,:,n)* will return the burn shape at index *n*. (You typically don't need to use *B*)

SA is a Tx2 array. *SA(:,1)* contains the $\prod_T$ vector while *SA(:,2)* contains the $\prod_A$ vector. *SA* can be useful because under "workspace" you can copy *SA* and import it into excel to scale the time and amplitude to useable units.

4. END-BURNER SIMULATION I/O

The *makeBore* function is not used when simulating an "end-burner." You can use the following command line:

$$\boldsymbol{[B\ SA]=burnArea(zeros(200,200),\ \prod_{L});} \qquad (5.1)$$

However, it is important to note that from the way the code is set up that the surface area value will be reported as exactly twice as high as it should be. Divide all of $\prod_A$'s values by two. In addition, the burn duration will be reported to be half what it really is. Multiply all $\prod_T$'s values by two. The duration of burn should be about *L/r*. It is still unclear whether this mode terminates the simulation properly. You can simply cut off the thrust curve at *t=L/r*. It is also simple and adequate to predict an end-burner's thrust curve by hand.

5. Sample Run

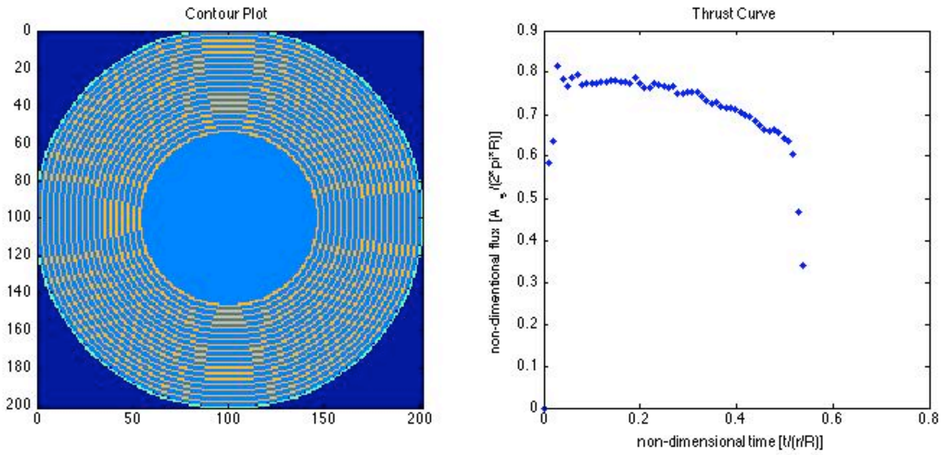
Suppose I have a fuel grain with measured grain diameter 1.1 inches, length 1.5 inches, and circular bore diameter 0.5 inches and a burn rate of 0.25 in/sec. Solving for $\prod_R$, $\prod_L$ I get:

$$\prod_R = 0.4545$$
$$\prod_L = 2.727$$

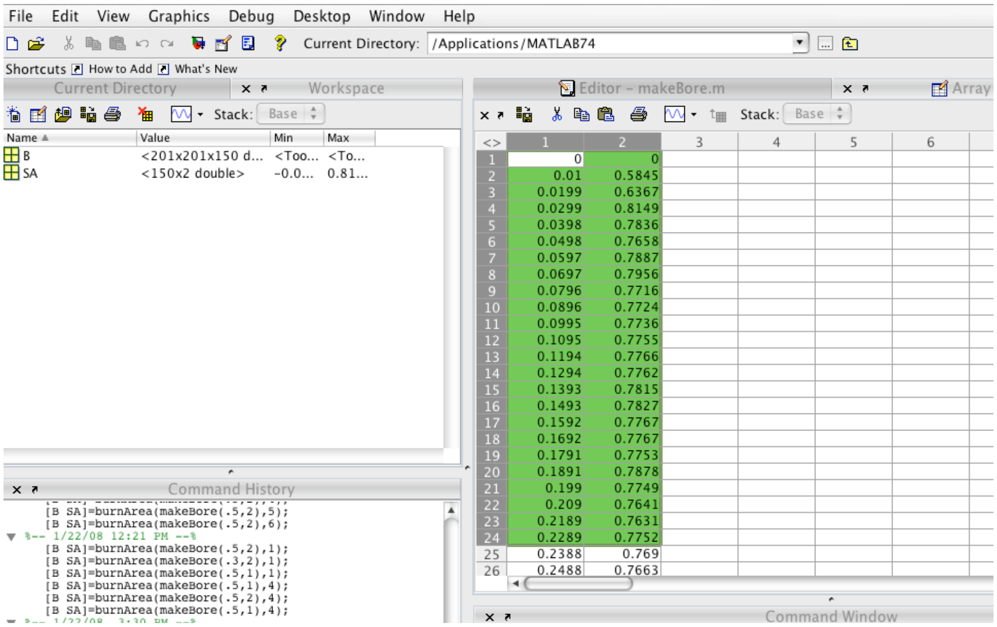
I enter this in Matlab's command line:

[B SA]=burnArea(makeBore(0.4545,2), 2.727);

Matlab returns an animation and the following display:



I access the data in the array SA via "Workspace":



Name	Value	Min	Max
B	<201x201x150 d...	<Too...	<To...
SA	<150x2 double>	-0.0...	0.81...

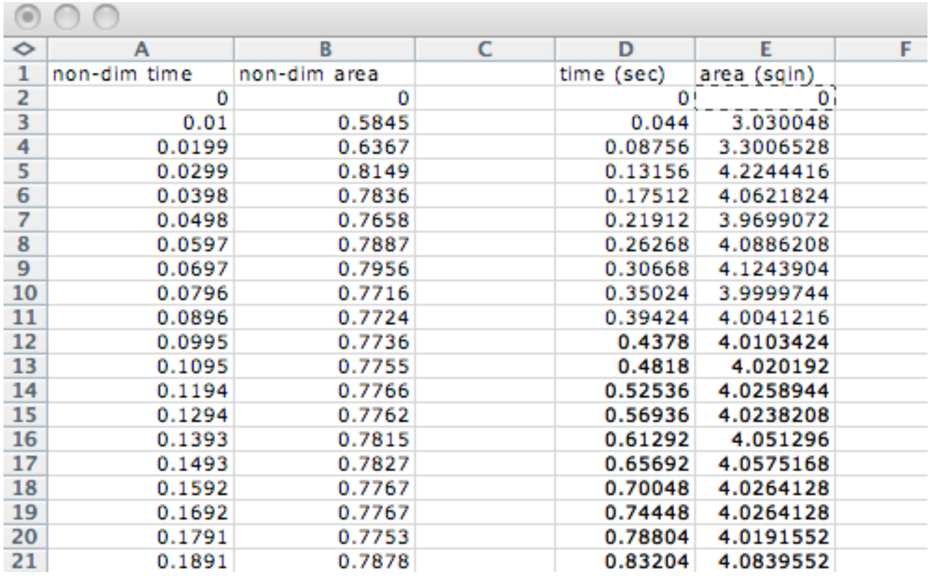
```
    [B SA]=burnArea(makeBore(.5,2),5);
    [B SA]=burnArea(makeBore(.5,2),6);
%-- 1/22/08 12:21 PM --%
    [B SA]=burnArea(makeBore(.5,2),1);
    [B SA]=burnArea(makeBore(.3,2),1);
    [B SA]=burnArea(makeBore(.5,1),1);
    [B SA]=burnArea(makeBore(.5,1),4);
    [B SA]=burnArea(makeBore(.5,2),4);
    [B SA]=burnArea(makeBore(.5,1),4);
%-- 1/22/08  3:30 PM --%
```

<>	1	2	3	4	5	6
1	0	0				
2	0.01	0.5845				
3	0.0199	0.6367				
4	0.0299	0.8149				
5	0.0398	0.7836				
6	0.0498	0.7658				
7	0.0597	0.7887				
8	0.0697	0.7956				
9	0.0796	0.7716				
10	0.0896	0.7724				
11	0.0995	0.7736				
12	0.1095	0.7755				
13	0.1194	0.7766				
14	0.1294	0.7762				
15	0.1393	0.7815				
16	0.1493	0.7827				
17	0.1592	0.7767				
18	0.1692	0.7767				
19	0.1791	0.7753				
20	0.1891	0.7878				
21	0.199	0.7749				
22	0.209	0.7641				
23	0.2189	0.7631				
24	0.2289	0.7752				
25	0.2388	0.769				
26	0.2488	0.7663				

I need to scale these numbers back to dimensional numbers. Because of this I calculate the scaling factors for my conditions:

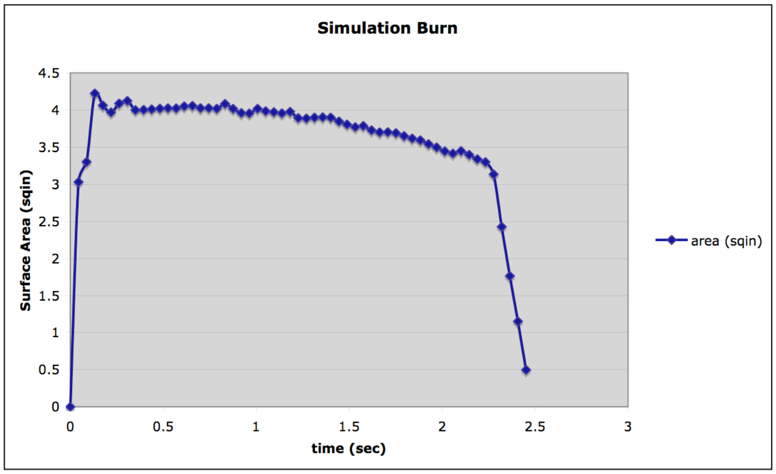
$$A=2\pi RL \prod_A \rightarrow A/\prod_A=2\pi RL = \mathbf{5.184}$$
$$t=\prod_T R/r \rightarrow t/\prod_T = R/r = \mathbf{4.4}$$

In Excel the data can easily be scaled:

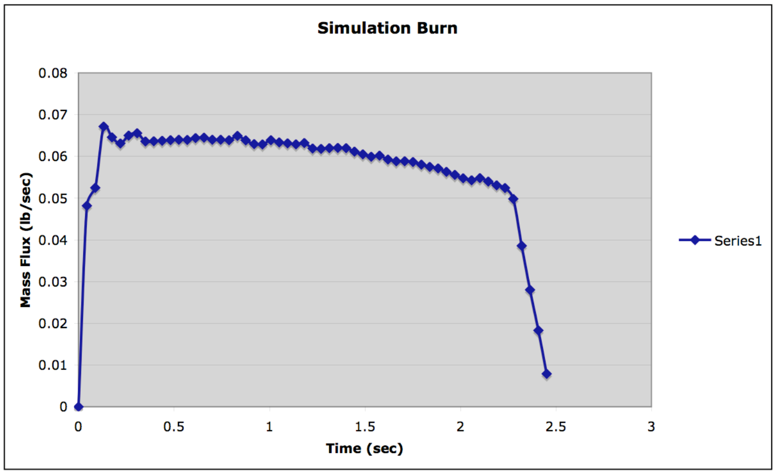


◇	A	B	C	D	E	F
1	non-dim time	non-dim area		time (sec)	area (sqin)	
2	0	0		0	0	
3	0.01	0.5845		0.044	3.030048	
4	0.0199	0.6367		0.08756	3.3006528	
5	0.0299	0.8149		0.13156	4.2244416	
6	0.0398	0.7836		0.17512	4.0621824	
7	0.0498	0.7658		0.21912	3.9699072	
8	0.0597	0.7887		0.26268	4.0886208	
9	0.0697	0.7956		0.30668	4.1243904	
10	0.0796	0.7716		0.35024	3.9999744	
11	0.0896	0.7724		0.39424	4.0041216	
12	0.0995	0.7736		0.4378	4.0103424	
13	0.1095	0.7755		0.4818	4.020192	
14	0.1194	0.7766		0.52536	4.0258944	
15	0.1294	0.7762		0.56936	4.0238208	
16	0.1393	0.7815		0.61292	4.051296	
17	0.1493	0.7827		0.65692	4.0575168	
18	0.1592	0.7767		0.70048	4.0264128	
19	0.1692	0.7767		0.74448	4.0264128	
20	0.1791	0.7753		0.78804	4.0191552	
21	0.1891	0.7878		0.83204	4.0839552	

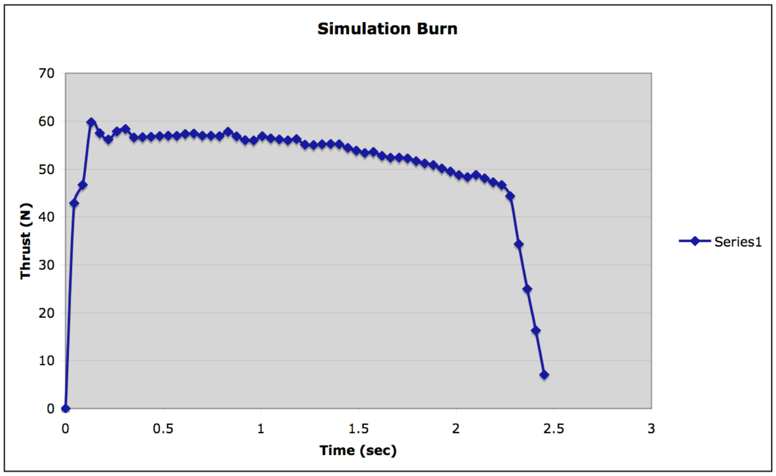
Finally, I can make a graph of the dimensional simulation data:



If desired, I can multiply the area by the burnrate and density (about 0.063 lb/in^3) of APCP to get mass flux through the nozzle:



Through some hoops and bounds, I can acquire a rough thrust curve:



Now, does my thrust curve seem reasonable given my conditions? I say yes. I can easily calculate the average thrust (51.5 N) and the total impulse (126 Ns). What other parameters could one extract?

6. Additional Notes

This simulator is designed to simulate single fuel grains. If multiple fuel grains are used, enter the specifications for one of the fuel grains and simply multiply the output (surface area, thrust, etc.) by the number of grains.

To acquire the linear burnrate *r*, examine the measured thrust curve and determine the burntime *T*, of the motor. If the motor is a circular bore, one can find the average burnrate by figuring how long it took to burn through the “web” of the propellant grain. This can be expressed mathematically as:

$$r = \frac{R - R_{bore}}{T} \qquad (7.1)$$

If the motor is an end-burner (such as the G69N), the average burnrate can be expressed as:

$$r = \frac{L}{T} \qquad (7.2)$$