# OpenArch: Guidance notes

Written by Tom McLean

March 17, 2021

## 1 Introduction

The OpenArch open-source package has 2 main folders: *Functions and Definitions* and *Elliptical example*. *Functions and Definitions* contains the key functions and class definitions for analysis or presentation MATLAB scripts. *Elliptical example* contains example scripts on how the files within *Functions and Definitions* should be used.

## 2 Glossary

The following terms may be useful in understanding the key inputs for the scripts.

- *BLOCK*: Structure containing the coordinates of one voussoir. Takes as input the polar coordinates of the intrados and extrados of the voussoir and automatically calculates centroids/ forces/ masses etc.
- *ARCH/geom (depending on script)*: Usually refers to an array of BLOCK structures. Use Toolkit.Build(ARCH)/ Toolkit.Build(geom) to plot the arch.
- *L*: Amount of additional mass distributed on top (L = 1 corresponds to the mass of the arch applied again, L = 0 corresponds to no additional mass)
- *alpha_m*: Inertial contribution of the additional mass (0-1)
- *StartingPoints*: Refers to the right hand side starting point for the thrust line (construction is from right to left). Can be given as an array or a single value. (Different options are sometimes commented out if you want to change them back)
- *xfinds*: Refers to the left hand side end point for the thrust line. Can be given as an array or a single value.
- *xp, yp*: Initial position of the pole of the force polygon – generally this is arbitrary and can be set to (0,0) for slightly quicker convergence. But at higher levels of loading may be necessary to place a large distance to the left to ensure it converges (this is how I have left the scripts).
- *tol*: Determines how accurate the bisection method to find the limit thrust line is once an admissible line has been found.

# 3 Standard Geometries

For each standard geometry (circular, parabolic, catenary) considered there should be two main scripts. One script will find the minimum thickness (i.e. EllipticalMinThickness.m), these scripts wont do any plotting and will simply return the minimum thickness value. The other scripts (i.e. EllipticalArches.m) will take in a given t/R and try to find an admissible thrust line for the given loading, and then give plots of the thrust line/ rupture angles etc.

The minimum thickness algorithm takes an overestimated thickness and reduces it until it obtains the minimum thickness for a given loading.

Instructions:

1. Input the loading conditions (gravity, lateral loading, overburden)
2. Give the algorithm a thickness to start with (ensure an overestimate). This could be taken from figures in the dissertation.
3. Decide the starting (right hand side) and end (left hand side) points. If the arch is known to be a two springing mechanism set these as single values rather than arrays for a faster calculation (StartingPoints = x_ext, xfinds = x_int). Note these are within the body of the script so you will have to find them.
    - Circular arch reaches two springing mechanism at approx. $\epsilon/\alpha = 0.6$ (where $\alpha$ is the gravitational multiplier)
    - Parabolic arch reaches two springing mechanism at approx. $\epsilon/\alpha = 0.1$ (where $\alpha$ is the gravitational multiplier)
    - Catenary arch always a two springing mechanism for any lateral load.
4. Run script and make note of the $t/R$.
5. Input the $t/R$ value into the plotting script, and ensure all the other parameters are set the same as previously before running it.
6. The plotting script will then save a .mat file containing a struct type file with the arch geometry and thrust line coordinates.
7. A presentation script can then be used using the saved .mat file as an input to generate an image of the thrust line and its corresponding collapse mechanism.

# 4 Worked example - Elliptical arch subject to $\epsilon = 0.3$

The open-source package includes example scripts for elliptical arches. Other geometries can be replicated from this script by simply changing how the geometry is created - this will not affect the analysis.

The first two analysis scripts, *EllipticalMinThickness.m* and *EllipticalArches.m* were created using the scripts within the *Analysis* folder in *'Functions and definitions'*:

*EllipticalMinThickness.m*:

1. Set global variables, e.g. GravitationalAcceleration, LateralAcceleration, Num_Blocks etc.
2. Set the increments to reduce thickness by (defines level of accuracy) e.g. $dts = [0.1, 0.01, 0.001]$ will find the minimum thickness to 3 decimal places.
3. Reduce thickness until no admissible thrust line can be found:
   (a) Generate new arch geometry for the new thickness
   (b) Apply relevant overburden if appropriate using *AddSoil.m*
   (c) Define *StartingPoints* (right hand side of thrust line, for the case fo $\epsilon = 0.3$ we know there is a hinge at the extrados of the right hand springing for left-to-right loading so it is set at this point)
   (d) Define end points (or *xfinds*) (Left hand side of thrust line. For the load considered, the end point of the thrust line will lie between the intrados and extrados so an array is given to allow the algorithm to search for a solution between the two extremes)
   (e) Define the position of the pole of the force polygon $(x_p, y_p)$ which is arbitrary. For large loads, this should be set far to the 'left' (e.g. 5000,5000) in order to ensure convergence.
   (f) Define the 'load line' $(x_{ll}, y_{ll})$ of the force polygon using the forces acting at the centroid of each voussoir.
   (g) Calculate an initial thrust line using $[x, y] = Toolkit.ThrustLine(ARCH, x_{ll}, y_{ll})$
   (h) For each defined end position (*xfinds*) run $[x, y, success] = PoleFinding.find\_minimum\_thrust(ARCH, x, y, x_{ll}, y_{ll}, xfind, tol)$. This determines if there is an admissible thrust line, and if so returns its coordinates.
   (i) At any point if an admissible thrust line is found the thickness is then reduced by the current thickness increment. If after all end points (*xfinds*) have been checked there is no admissible thrust line then the arch is too thin, and the thickness is set to its immediately previous value and the next thickness increment is selected.
   (j) Repeat process with the next thickness increment until all thickness increments have been used.
4. Display the final value of $t/R$

*EllipticalArches.m*:

1. Script almost identical to *EllipticalMinThickness.m*, but takes as input the result from that script for the value of $t/R$.
2. Calculate the thrust line as before.
3. Use $hinges = Toolkit.HingeLocations(ARCH, x, y, 5, "extrados")$ (indicating that the first hinge on right hand side is expected at extrados) to find the hinge locations.
4. Plot the the results as given in Figures 1 & 2
5. Save the results into the a struct file for use at later date/ in presentation scripts if required.

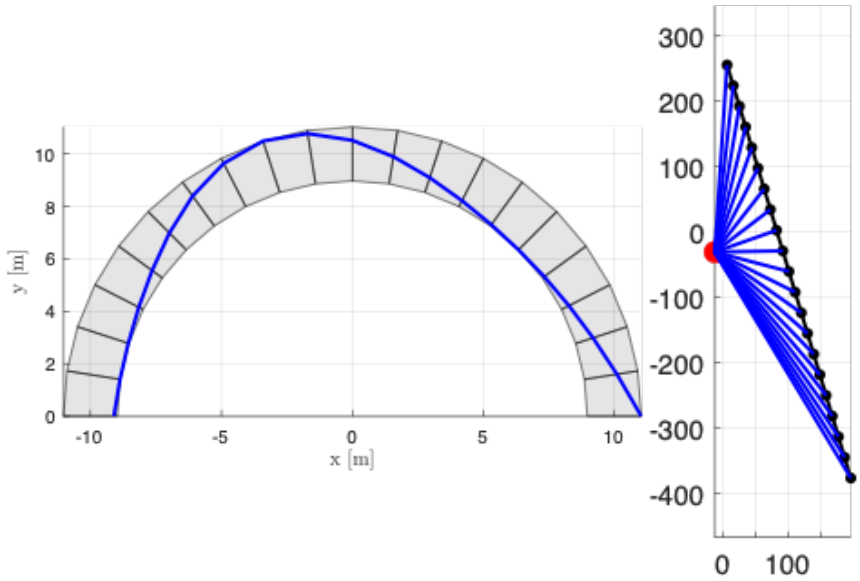


Figure 1: First plot output: Limit thrust line & voussoir edges alongside its corresponding force polygon

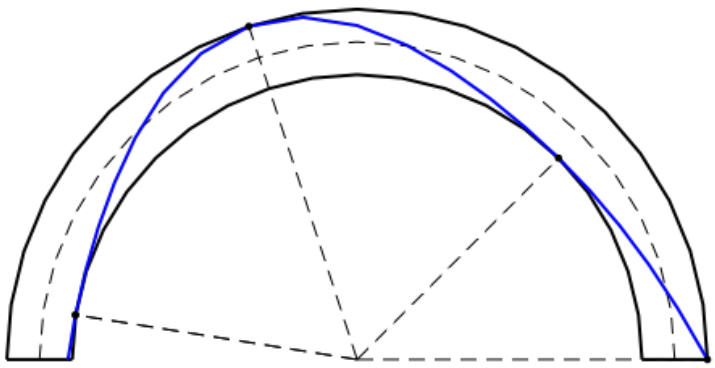

Figure 2: Second plot output: Rupture angles and limit thrust line