

# NACA 4-Digit Airfoils for Stunters, Part I

## The Lost Byte Method of Making Wing Rib Templates

by Larry Cunningham

### NACA 4-Digit Airfoil Generator

Although the **NACA 4-digit airfoil generator function** was clearly not created for model airplane use, it works rather well. Constrained to producing the symmetrical airfoils used for control line stunters, the generator function is reduced to a simple polynomial equation:

$$y = \pm 0.5(A\sqrt{x} + Bx - Cx^2 + Dx^3 + Ex^4)$$

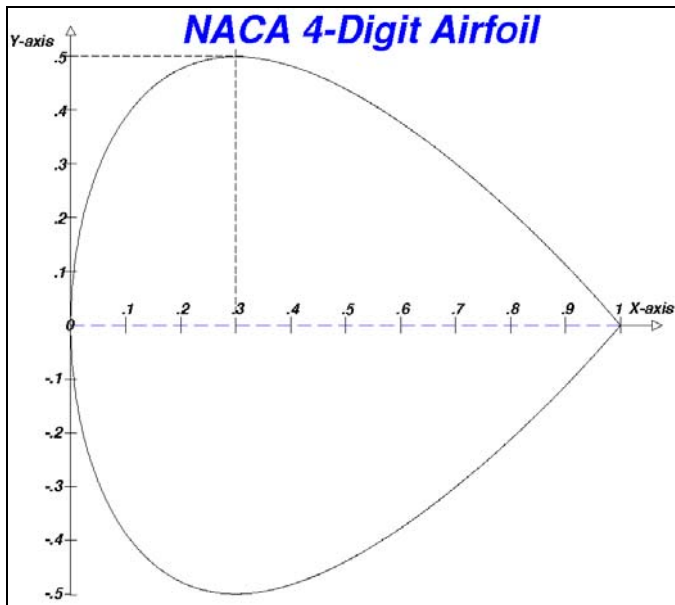
where **y** is the vertical coordinate  
**x** is the horizontal coordinate

$$0 \leq x \leq 1$$

and the coefficients are

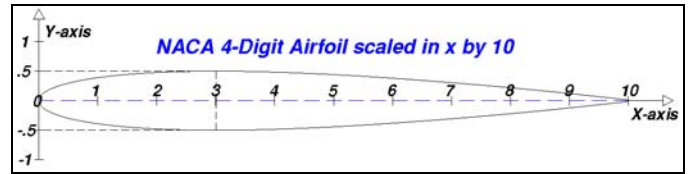
<b>A</b> = 0.2969
<b>B</b> = -0.1260
<b>C</b> = -0.3516
<b>D</b> = 0.2843
<b>E</b> = -0.1015

The output **y** corresponds to a **pair** of symmetrical vertical coordinates (**+y** and **-y**) for a **normalized** airfoil, whose chord length in **x** is exactly 1 unit long. The leading edge of the airfoil is located at **(x,y)** of **(0,0)**, and its trailing edge is at **(1,0)**.

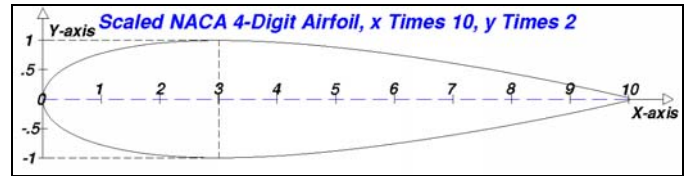


A characteristic of the NACA 4-digit airfoil is that its maximum thickness occurs at **30%** of the chord length from the leading edge, at **x = 0.3** on the normalized airfoil. The maximum total thickness is also exactly 1 unit high, and occurs at **(0.3, 0.5)** and **(0.3, -0.5)**. This makes the normalized airfoil shape rather **FAT**, with a full 100% thickness!

To obtain coordinates for a useful airfoil, the normalized airfoil data must be **scaled**. First, we have to consider the **chord** (horizontal **x** length) of the desired airfoil. For example, if our actual airfoil is to be 10 inches long, we must multiply all **x** coordinates by the actual chord length, 10 inches. This produces an airfoil that is 10 inches long and 1 inch thick.



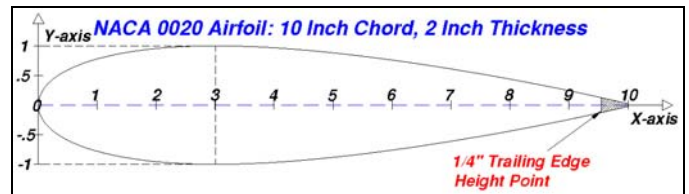
Maximum thickness is the other scaling factor, to be applied to the vertical **y** coordinates only. For example, if we require a maximum thickness of 2 inches, all **y** coordinates must be multiplied by 2 inches.



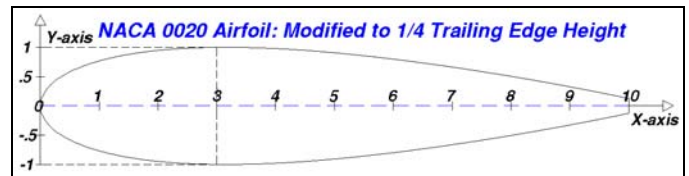
For symmetrical airfoils, the NACA 4-digit airfoils use the numbering **NACA 00nn**, where **nn** is a 2-digit number representing maximum thickness as a percentage of chord. The example airfoil which is 2 inches thick and 10 inches long is a 20% thickness, so its designation would be **NACA 0020**. Simple enough so far? Clearly the airfoil is in control-line stunt territory here!

Another consideration for practical use on stunters is an adjustment for trailing edge height. The problem is that the NACA 4-digit airfoil comes to a nice point, with zero height, while most stunter wing trailing edges will accommodate the leading edge of a flap, typically 3/16 to 3/8 inch thick.

The approach of simply "chopping off" the trailing edge of the NACA 4-digit airfoil where its thickness is the desired height works well. The first step is to locate the **x** position where this occurs on the scaled airfoil. For example, on the **NACA 0020** airfoil with a 10 inch chord length, the trailing edge height is 1/4" at an **x** position of 9.5427 inches. (I did not determine this by solving the 5th order polynomial NACA 4-digit generator equation! This position was located by iteration.)



To rescale the chord to produce a 10 inch long airfoil with a 1/4" high trailing edge, we can simply divide all the **x** coordinates by the **x** position 9.5427 and multiply by 10.



# NACA 4-Digit Airfoils for Stunters, Part I

## The Lost Byte Method of Making Wing Rib Templates

by Larry Cunningham

We can then simply constrain the scaled  $x$  range from 0 through 10 inches (finished chord length) to effectively ignore the "chopped off" part.

Finally, to clean things up and actually "close" the trailing edge of the scaled airfoil shape, return one additional coordinate for the trailing edge position, e.g. (10,0) in the example. We now have a set of coordinates we can plot!

### A Generalized NACA 00nn Airfoil Function

Based on my description, many readers could write a simple computer program to generate a plottable set of  $x$  and  $y$  coordinates for a symmetrical NACA 4-digit airfoil, properly scaled and adjusted for a non-zero trailing edge height as needed for the stunt application.

Herein I will define a computing function for the NACA 00nn airfoil discussed above. I seek a generalized computing "object", a magic "box" named *Airfoil*, which accepts certain [minimal] information and returns coordinates for the airfoil contour. The output of this object will be a point on the airfoil contour, defined by the scaled coordinates (x,y). Since our airfoil is symmetrical about the X-axis, the opposite point on the airfoil contour is simply (x,-y).

What is the minimum input to such a computing object? Clearly, it needs to "know" the scaled length, thickness, and trailing edge height of the target airfoil.

One might suppose that *Airfoil* also needs to know the scaled  $x$  value of the point (x,y), so that it can compute the scaled  $y$  value! But I am looking for a scaled point (x,y) output, because I wish to query *Airfoil* in a specific, useful manner.

Instead of  $x$ , I will specify a total number of points on my airfoil contour plot, and the index of the point (x,y) I wish to query. For example, my airfoil plot may have 2000 points, evenly spaced along its chord (horizontal X-axis), and I want a scaled point value (x,y) returned for the point whose index [1..2000], is, say 273. I want *Airfoil* to take care of all the scaling and determine  $x$  from the point number information.

Each time I query *Airfoil*, I will supply the following information:

<b>X</b>	scaled chord length of the finished airfoil
<b>T</b>	maximum thickness of the finished airfoil
<b>H</b>	trailing edge height of the finished airfoil
<b>P</b>	the total number of evenly spaced points in my plot
<b>N</b>	the index of a point (x,y) I am querying

I want *Airfoil* to return a single point (x,y):

<b>x</b>	scaled horizontal position of the specified point
<b>y</b>	scaled vertical position of the specified point

I define *Airfoil* mathematically as follows:

$$(x,y) = \textit{Airfoil}(X,T,H,P,N)$$

This may seem a bit strange. You may be wondering "Why query only a single point? You supply all the information needed, why not simply compute a whole set of points?" Bear with me, my motives will soon be made clear shortly.

### Something Useful

Seeing as how we are usually interested in a whole set of rib templates for a complete wing panel, we need to tackle the airfoil contour problem in a general manner.

Consider interpolation. Specifically, suppose we define a wing panel with two airfoils: one for the **Root** of the wing and the other for the **Tip**. Unless we plan to build straight wings of constant thickness, we will be interested in a general case where the root and tip airfoils can be different, and all the ribs in between them have a shape which is interpolated.

### A Mathematical Foam Cutter

The *Airfoil* object as just described, calculates and returns (x,y) point coordinates of any [symmetrical, modified NACA 4-digit] airfoil I am interested in. For example, suppose I was working on a wing with the following root and tip airfoils:

<b>Root:</b>	X=10	scaled chord length
	T=2	maximum thickness
	H=0.25	trailing edge height
<b>Tip:</b>	X=8	scaled chord length
	T=1.5	maximum thickness
	H=0.25	trailing edge height

If I wanted the **192nd** point of 10,000 points for Root and Tip airfoils, I could call *Airfoil* to get the point for each plot:

<b>Root:</b>	(x,y) = <i>Airfoil</i> (10,2,0.25,10000,192)
<b>Tip:</b>	(x,y) = <i>Airfoil</i> (8,1.5,0.25,10000,192)

Can you see where I am headed, here? The scaled coordinates of the point (x,y) output by the *Airfoil* object are always defined in a little "local" coordinate system for the rib template we want to plot; the origin of that coordinate system is always at (0,0), the point which is the leading edge of our airfoil.

Now, everything is wonderful until I want points for ribs other than the root and tip, which the *Airfoil* object cannot provide directly. Clearly, for these points, we need interpolation!

But what gets interpolated? Obviously, the  $y$  coordinate, airfoil height has to be interpolated. What about its corresponding  $x$  value? Remember that we are dealing with the local coordinate system for each rib. Can you see the complication that could easily slip into our interpolation if we try to deal with the *Airfoil* object by supplying  $x$  positions on the root and tip ribs? We would have to deal with **three** local airfoil template coordinate systems (for the root, the tip, and the rib we are trying to interpolate).

Instead, try to look at the problem in a simpler, more practical way. As a strong analogy, consider how we must guide a hot wire as it cuts through a foam wing. Ideally it needs to track the chords of both the root and tip templates at an even **percentage** rate, on **both** ends. For example, as the hot wire moved, it would be nice if it was at the 10% chord position on both the root and tip templates at the same time. And continue smoothly, finally reaching 100% of chord position (the trailing edge) at exactly the same time.

In fact, when you think about it, you realize it is **imperative** that the cutting wire moves through the foam wing in this manner,

# NACA 4-Digit Airfoils for Stunters, Part I

## The Lost Byte Method of Making Wing Rib Templates

by Larry Cunningham

otherwise the proper (interpolated) contour of the wing is **guaranteed** to be distorted!

Hopefully my motive for defining the *Airfoil* object becomes clear now. As a practical matter, I probably also want to plot the same number of points for each airfoil contour: P. And I want to know the scaled coordinates (x,y) for each of these points. Clearly, I wish to run a purely abstract, **mathematical foam cutter** with the same motion as a hot wire! And doesn't it all fit so nicely with my *Airfoil* object?

### Darr, Linear Interpolation

Linear interpolation involves locating a point on a straight line between the two end points. In this case, the straight line runs between our root and tip airfoil contour surfaces, at some percentage of their chord lengths.

A simple general equation for linear interpolation is:

$$V = R + (Z/L)(T - R)$$

- where
- V** = interpolated value
  - R** = value at Root
  - T** = value at Tip
  - Z** = rib distance from Root
  - L** = Root to Tip length

The quantity **Z/L** can also be thought of as a normalized distance **D**, that is **D = Z/L**.

### An Interpolator Object

You guessed it. My next step is to define another computing object, this time to compute interpolations of *Airfoil* point coordinates. Call it *Interpolate*. What might we expect to feed it? Consider:

- Xr** scaled chord length at Root
- Xt** scaled chord length at Tip
- Tr** maximum thickness at Root
- Tt** maximum thickness at Tip
- D** normalized distance from Root to desired rib
- H** trailing edge height of the finished airfoil
- P** total number of evenly spaced points in my plot
- N** index of the point (x,y) that I am querying

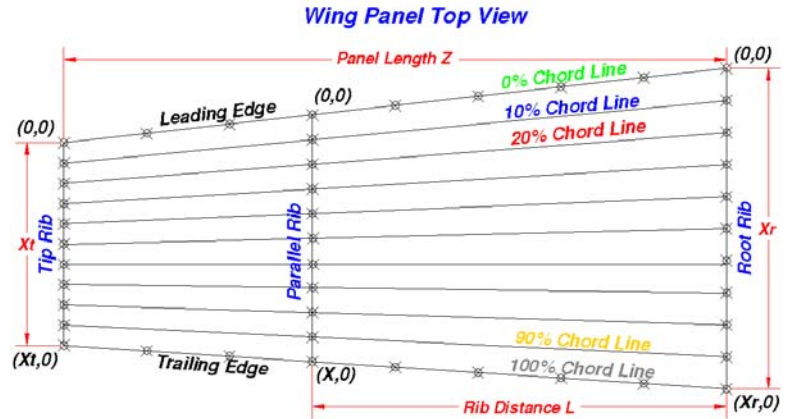
Again, I am interested in a scaled point (x,y) as an output from my Interpolator, which I will define mathematically:

$$(x,y) = \text{Interpolate}(Xr, Xt, Tr, Tt, D, H, P, N)$$

Now we are in business. The cool thing about *Interpolate* is how easy the calculation is. Simple query *Airfoil* for the Root and Tip scaled coordinates (x,y) and apply our simple interpolation function to both **x** and **y** values!

And it works perfectly. It may seem a bit strange to interpolated scaled local coordinates for the root and tips in both axes and get scaled local coordinates for a particular rib position. But it is mathematically correct.

The application to "straight ahead" ribs is illustrated here:



Linear Interpolation and Parallel Ribs

### But You Wanted "Geodesic" Ribs!

Of course you did! Free flight technology from the 1940's is now being brought to bear on stunter wing construction, in the interest of optimizing strength and minimizing weight (grin!).

Suddenly we all need "geodesic" ribs, which lie diagonally in the wing panel instead of parallel to the root airfoil.

Part 2 of this article series will address complications that this introduces into the interpolation calculation. In addition, I will enhance my mathematical foam cutter to do things outside the capabilities of physical hot wires: elliptical interpolation of wing thickness!

I will also discuss some additional tasks we might put the computer to work on, such as placement of spar shapes within the rib airfoil contour. I have developed a little PC-compatible MSDOS program named *StuntRib*, which outputs DXF graphics files that can be imported into most CAD programs.

Sample rib templates for my *Mo'Best* stunter, from airfoil contours generated by *StuntRib* are illustrated below. *StuntRib* is available to interested PAMPA members as a free download from the PAMPA FTP web site at <ftp://pampaftp.pampa@ftp.zianet.com>



Airfoil contours generated by *StuntRib*

-Larry Cunningham