BLENDER AIRCRAFT MODULE

Team Members: Al Ang, Allison Redderson Lear, Brendan Hickey, Michael Ding, Maaz Syed, Robinne Ponty
Advisors: Dr. Farzad Ahmadkhanlou, Prof. Vince McDonell *Company Liaison:* J. Phillip Barnes

Background & Requirements

Sponsored by Northrop Grumman Corporation, the project goal is to create an aircraft modeling software that will be integrated into a comprehensive program that tests dynamic conditions of a user defined airplane.

- Utilize multiple calculation methods with varying speed and precision
- Parametrically define curves and surfaces
- Accessible user interface
- Use of user-defined aircraft constants to determine aircraft geometry

Project Goals & Impact

- Create modeling software in Blender using Python
- Allow user-defined aircraft parameters
- Final product will be an intuitive program that can be used to model aircraft quickly and accurately, with potential for rendering the final model for display, simulation, and 3D printing.

- Allow engineers to test new hypothetical aircraft structures and geometries for new designs
- Will be widely available and virtually free to run due to open source Blender 3D software

Current Progress & Future Development

- Fully functional trigonometric equation-based model generation, complete with working User Interface
- Functional Cubic Spline generation of wing and fuselage geometries
- Generation of 2-Dimensional curves using Parametric Polynomials
- Rendering and texturing capabilities

- Build Polynomial 2-D curve into 3-D mesh generating method
- Integrate Cubic Spline, Parametric Polynomial, and rendering techniques into comprehensive UI

Mathematical Theory

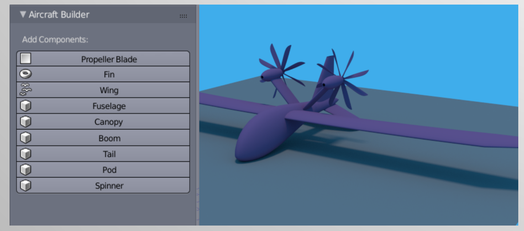
Trigonometric Functions
Polynomial Functions
Cubic Spline

Programming

Python

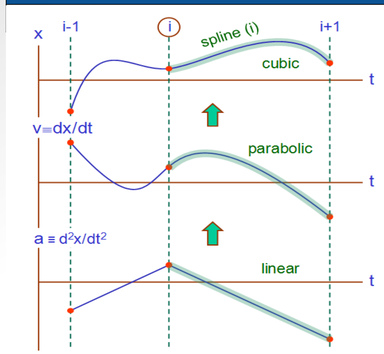
Modeling Software

Blender 3D



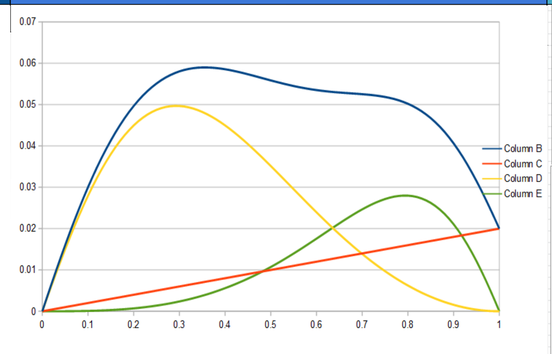
Sample rendering of trigonometrically modeled propeller plane with user interface

Parametric Cubic Splines



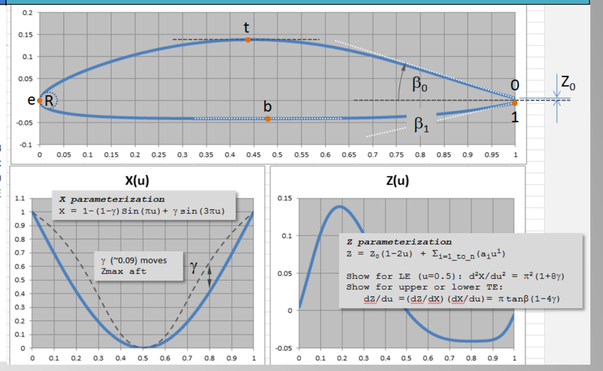
Visualization of a cubic spline and its derivatives

Parametric Trigonometric Functions



The curve of a Boeing 747 fuselage (blue) built as the sum of 3 trigonometric functions

Parametric Polynomial Functions



A sample airfoil characterized by important points, slopes, and a radius, and parameterized by "u" variable

```
file = csv.reader(open('\\Users\Michael Ding\BlenderGridInputPhilAlpha.csv', newline=
the file

#curRow = [] # placeholder variable, denoting current row

#
def fuselage():
#    close approximation of cubic-spline model using Phil's trigonometric functions
    global numu, numv, verts, rqd, dqr, piq2 #more Phil-generated variables
    curRow = [] # curRow is the pointer that points to which row I want the index to
    for idx, row in enumerate(file): #idx seems to be index, and ticks up each time t
ticks once, as seen in the next line.
        if idx<1:
            curRow = row #assigns curRow to the row assigned by idk
            scale_input = str(curRow[0]) # the following are created by me and are re
be apparent which variables go with which.
            dze_input = str(curRow[1])
            v_input1 = str(curRow[2])
            v_input2 = str(curRow[3])
            w_input = str(curRow[4])
            zu_input = str(curRow[5])
            ze_input = str(curRow[6])
            zL_input = str(curRow[7])
            x_input = str(curRow[8])
            u_input = str(curRow[9])
            y_input = str(curRow[10])
            z_input = str(curRow[11])

            # mesh variables
            numu = 15 ; numv = 31 ; scale = eval(scale_input) ; dze=eval(dze_input)
            # fill verts array, using "1-to-N" subscripting and smart spacing
            j = 1
            while j<=numv: #
                v1 = eval(v_input1)
                v = eval(v_input2) # finer ends spacing
                w = eval(w_input)
                zu = eval(zu_input)
                ze = eval(ze_input)
```

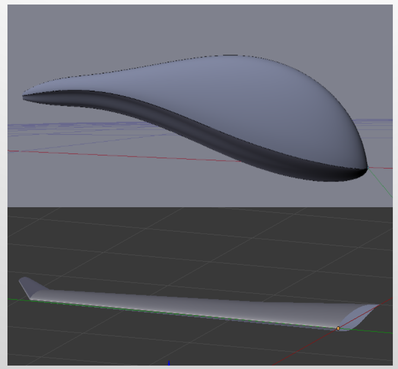
```
j = 1
while j<=numv: #
    v1 = eval(v_input1)
    v = eval(v_input2) # finer ends spacing
    w = eval(w_input)
    zu = eval(zu_input)
    ze = eval(ze_input)
    zL = eval(zL_input)
    x = eval(x_input)
    hu = zu-ze ; hL = ze-zL
    i = 1
    while i<=numu:
        u = eval(u_input)
        y = eval(y_input)
        z =eval(z_input)
        vert = (-4+scale*x, scale*y, -1+scal
```

```
v1-0.15*sin(2.*pi*v1)
0.001 + 0.0931 * sin(pi * (1 - v) ** 1.2) - 0.0171 * sin(pi * (0 + v) ** 1.2) + 0.0201 * sin(
0.001 + dze * v + 0.142 * sin(pi * (1 - v) ** 1.25) + 0.028 * sin(pi * (0 + v) ** 9.00)
dze * (sin(piq2 * v)) ** 2.90 + 0.009 * sin(pi * (0+v) ** 2.0)
(-1)*0.001 + dze * v - 0.0851 * sin(pi * (1 - v) ** 1.3) - 0.0201 * sin(pi * (0 + v) ** 7.0)
v**1.67
(i-1)/(numu-1)
w * cos(-piq2 + u*pi)
ze + (bool(u<0.5)*hL + bool(u>=0.5)*hu) * sin(-piq2 + u*pi)
```

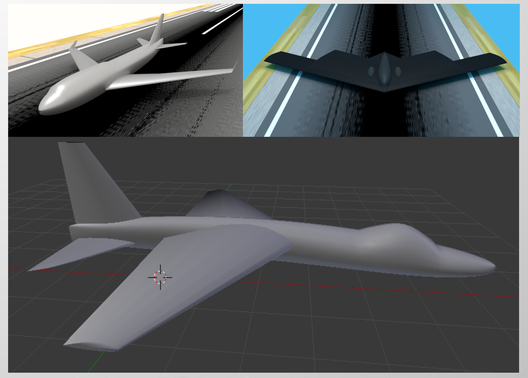
Code for building a fuselage using the parametric trigonometric method (left); Spreadsheet-formatted parameters (bottom right) that are input by the user and integrated into parametric polynomial code (top right)

```
def parametric_poly():
    beta_0 = 15*(pi/180)          #afterbody angle of upper trailing edge
    beta_1 = (0)*(pi/180)         #afterbody angle of lower trailing edge
    R = 0.015                     #normalized leading edge radius, radius/chord length
    u_e = 0.5                     #u at leading edge
    gamma = 0.09                  #X(u) graph shift
    u_0 = 0                       #u at upper trailing edge
    u_1 = 1                       #u at lower trailing edge
    u_t = 0.24                     #u at maximum height
    z_t = 0.14                     #maximum height
    u_b = 0.77                     #u at minimum height
    z_b = -0.04                    #minimum height
    z_0 = 0.005                    #upper trailing edge thickness
    z_1 = -z_0                     #height at lower trailing edge

    print(2**2*3)
    print(tan(beta_0))

    BB = [
            (pi*tan(beta_0)*(1-4*gamma))+2*z_0,          #upper trailing edge slope equation
            z_t-(z_0*(1-2*u_t)),                         #Maximum Z position equation
            2*z_0,                                       #Maximum Z slope equation
            0,                                           #leading edge position equation
            -((R*((pi**2)*(1+8*gamma)))**0.5)+2*z_0,     #leading edge radius equation
            z_b-(z_0*(1-2*u_b)),                         #Minimum Z position equation
            2*z_0,                                       #Minimum Z slope equation
            z_1+ z_0,                                    #Lower trailing edge position equation
            (pi*tan(beta_1)*(1-4*gamma))+2*z_0           #Lower trailing edge slope equation
          ]

    FF = [
            [1, 0, 0, 0, 0, 0, 0, 0, 0],                                          #upper trailing edge slope equation
            [u_t**1, u_t**2, u_t**3, u_t**4, u_t**5, u_t**6 ,u_t**7, u_t**8, u_t**9],   #Maximum Z position equation
            [1, 2* u_t, 3*u_t**2, 4*u_t**3, 5*u_t**4, 6*u_t**5, 7*u_t**6, 8*u_t**7, 9*u_t**8],   #Maximum Z slope equation
            [u_e**1, u_e**2, u_e**3, u_e**4, u_e**5, u_e**6 ,u_e**7, u_e**8, u_e**9],   #leading edge position equation
            [1, 2* u_e, 3*u_e**2, 4*u_e**3, 5*u_e**4, 6*u_e**5, 7*u_e**6, 8*u_e**7, 9*u_e**8],   #leading edge radius equation
            [u_b**1, u_b**2, u_b**3, u_b**4, u_b**5, u_b**6 ,u_b**7, u_b**8, u_b**9 ],   #Minimum Z position equation
            [1, 2* u_b, 3*u_b**2, 4*u_b**3, 5*u_b**4, 6*u_b**5, 7*u_b**6, 8*u_b**7, 9*u_b**8],   #Minimum Z slope equation
            [1, 1, 1, 1, 1, 1, 1, 1, 1],                                          #lower trailing edge position equation
            [1, 2, 3, 4, 5, 6, 7, 8, 9]                                           #lower trailing edge slope equation
          ]
```

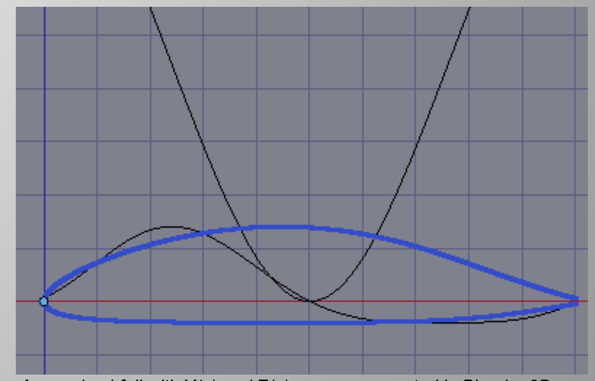
Sample code calculating the parametric polynomial of an airfoil



A cubic spline-generated fuselage (top) and wing (bottom)



Models of a Boeing 747 (top left), a B-2 bomber (top right), and a fighter jet, all built with trigonometric functions



A sample airfoil with X(u) and Z(u) curves, generated in Blender 3D using Parametric Polynomial Method