ENGR 105 – Introduction to Scientific Computing

Assignment 8

Due by 11:59 pm on Weds. 3/16/2022 on Gradescope

Problem 1 (15 pts): Legendre polynomials are solutions to Legendre's differential equation and are employed in some engineering fields including mechanics, fluid dynamics, thermodynamics, and electromagnetism.

The general recurrence formula for the Legendre polynomial may be mathematically represented as the following.

$$P_n(x) = \frac{1}{n}\left[(2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x)\right]$$

The first two Legendre polynomials are defined as $P_0(x) = 1$ and $P_1(x) = x$.

Create a recursive function that generates Legendre polynomials for any given positive, integer, and scalar `n` and scalar `x`. Your function should have the following declaration.

```
function Lp = p(n,x)
```

Use your function to compute `p(0,x)`, `p(1,x)`, `p(2,x)`, `p(3,x)`, etc. for a few values of *x*. `p(0,x)` should always yield 1, `p(1,x)` should always yield *x*, and you should compare your results with those of the known, analytic form for several cases of 0≤n≤6. This website contains analytical solutions to $P_0(x)$ through $P_6(x)$ that you may find helpful when evaluating the accuracy of your function.

Upload the function. Identify the name of the function and report the results of your function tests in your README.txt.

Problem 2 (25 pts): Measurements of the population of the United States for every decade from 1790 to 1950 in units of 1000 are:

```
[3939, 5308, 7240, 9638, 12866, 17069, 23192, 31443, 38558, 50156,
62948, 75995, 91972, 105711, 122775, 131669, 150697]
```

The following logistic model provides a good representation of this data set where *t* is the year.

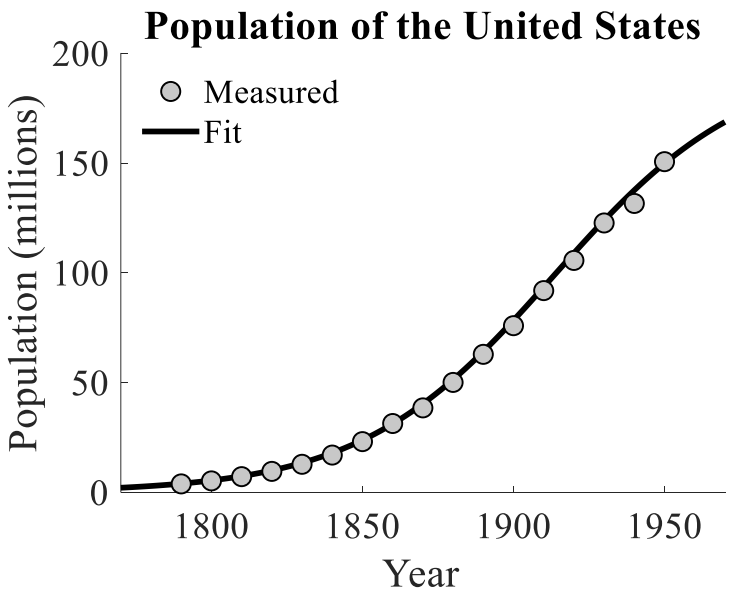
$$P(t) = \frac{197{,}273{,}000}{1 + e^{-0.03134(t-1913.25)}}$$

Produce a pretty plot of the population as a function of year with the data superimposed over the logistic population model using a script. The model series should extend -20 years and +20

years from the smallest and largest domain values of the measured data, respectively. All plot attributes should be modified using code (not via UI) and should include the following:

- The two plot series should be overlayed using independent `plot()` calls
- Circular data markers should be employed for the measured data – show influence over the marker size, marker face color specifically using RGB color values, and marker line color and size
- A solid line for the fit that shows influence over the line color and size
- Axes labels with influence over the font type and size
- A plot title with influence over the font type and size
- A legend in the northwest corner of the plot with influence over font type and size
- Removal of the top and right axes borders
- Background of the figure set to white

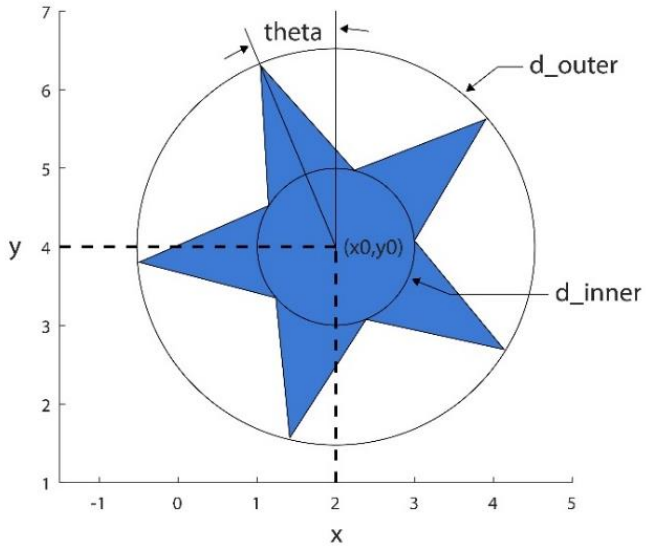
Your plot might look like the following.



You may employ older `set()`-based and/or more modern graphics handle/property methods to adjust the visual aspects of the plot.

Upload the script that produces the plot and a .jpg copy of the resulting plot. Identify the names of all files associated with this problem in your README.txt.

Problem 3 (25 pts): Create a function that plots a pentagram (*i.e*. a star) like the following using a `patch()` object.

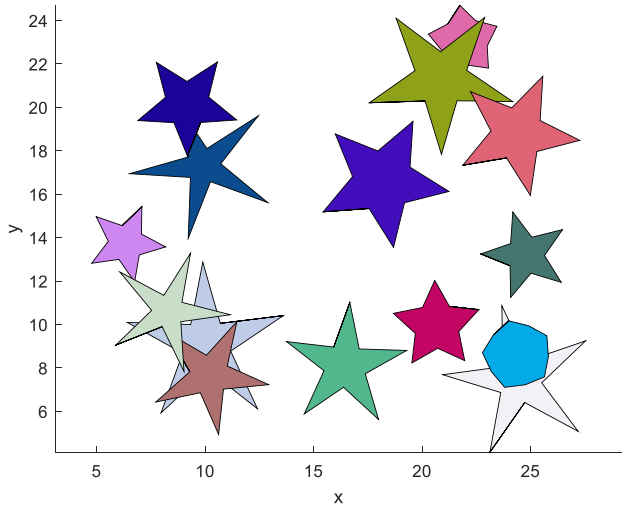


Your function should have the following function declaration.

```
function fivePointedStar(x0,y0,d_inner,d_outer,theta,c)
```

Input `c` should anticipate a 1 x 3 vector that contains the RGB color components of the `patch()` object. For example, the above was produced with `c = [.24 .48 .84]`. Input `theta` should have units of radians. Inputs `x0`, `y0`, `d_inner`, `d_outer`, and `theta` are scalars.

Create a script that calls on your `fivePointedStar()` function providing random locations (`x0` and `y0`), random diameters (`d_inner` and `d_outer`), and random colors (`c`) to produce least 10 stars. The following is an example plot with 15 stars.



Upload the `fivePointedStar` function, the test script, and a .jpg copy of the resulting test plot. Identify the names of all files associated with this problem in your README.txt.

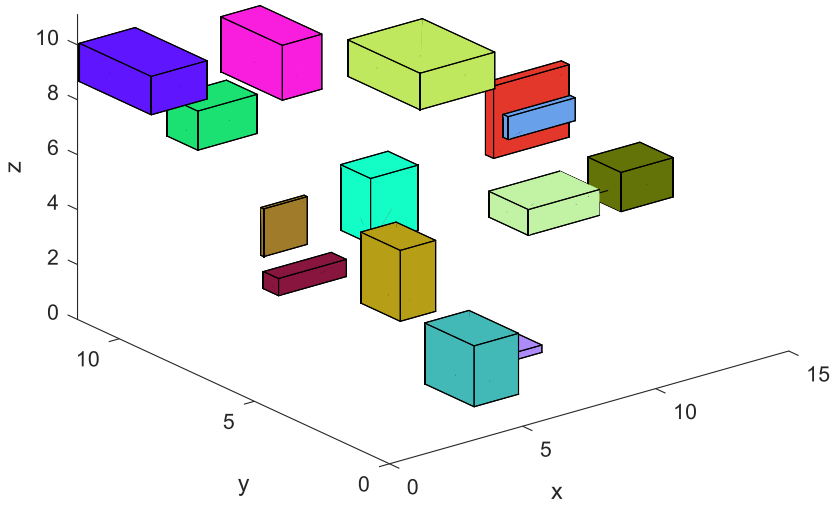
Problem 4 (25 pts): Create a function that plots a three dimensional closed single right rectangular prism with boundaries created from `fill3()` polygons. Use the following function declaration.

```
function rightRectPrism(x0,y0,z0,W,L,T,c)
```

Inputs `x0`, `y0`, and `z0` represent the lower corner of the prism in the *x*, *y*, and *z* axes, respectively. Inputs `W`, `L`, and `T` represent the lengths of the prism in the *x*, *y*, and *z* axes, respectively. Inputs `x0`, `y0`, `z0`, `W`, `L`, and `T` are scalars. Input `c` should anticipate a 1 x 3 vector that contains the RGB color components of the `fill3()` polygon.

Create a script that calls on your `rightRectPrism()` function providing random locations (`x0`, `y0`, and `z0`), random lengths (`W`, `L`, and `T`), and random colors (`c`) to produce least 10 right rectangular prisms.

The following is an example plot with 15 right rectangular prisms.



Upload the `rightRectPrism` function, the test script, and a .jpg copy of the resulting test plot. Identify the names of all files associated with this problem in your README.txt.