Outline: the following functions appear in the program in the same order they appear here.

From #pragma region to #pragma endregion is all setting up the GUI. It is advised not to alter this unless you are familiar with Windows forms.

**InitializeCostParameters** is used to initialize the component costs. It runs when the program is started. It fills out the component cost table with default values. Therefore it is rather simple to change these default values.

**populateSpline2V** takes a series of points (X,Y) and produces a matrix describing a cubic spline interpolation of these points.

**interpolateSplineV** takes an interpolation matrix (made by the previous function populateSpline2V) and the point to interpolate at, X, and returns a Y.

**interpolate2dSpline** takes a series of interpolation matrixes and a series of values and uses it to interpolate a value as a function of two variables, ie Z as a function of Y and X. in the program it is used to interpolate voltage as a function of SOC and current density.

**fileRead** reads a csv file and interprets it as a matrix and returns it. Used to read a matrix of V as function of current density and SOC and use cubic spline interpolation.

**Asinh** is the inverse hyperbolic sine, described here as it does not exist in cmath library.

**Voltage** is the electrochemical model, it returns a voltage as a function of SOC, cell area, flow rate, current density, aspect ratio of cell, cell chemistry, membrane used, and interpolation matrixes (optional). It currently contains two V-V electrochemical models, a Fe-V model, and a Li+ model. More may be added.

**CostTab** is used to read cost data from the component cost table.

**unitFunction** is an experimental function used for giving component costs as functions rather than constants.

**powerCost** calculates the stack and pump costs as a function of cell area, maximum flow rate, aspect ratio, cells per stack, power rating, system chemistry, and membrane type.

**energyCost** calculates the chemical and tank costs as a function of average effective voltage, depth of discharge, concentration, and system chemistry. The cost returned is per kWh.

**shuntLoss** calculates the shunt current power loss, in watts, as a function of cell voltage, area, channel dimensions, aspect ratio, and number of cells. It uses a finite different method to calculate this shunt power loss.

**pLoss** calculates the power loss from pumping electrolyte through the stack. Function of cell area, flow rate, aspect ratio, channel dimensions, number of channels. Uses darcy’s law to calculate pressure drop through cell, and hagen-poiseuille equation to calculate pressure drop through the cell channels.

**MarshalString** is used to convert managed strings to unmanaged strings – this does not need altered.

**slope** is a simple function used to calculate the slope between two points. Useful shortcut.

**calculate\_Click** is what executes when the calculate button is clicked. This can be thought of as the “main program”. This function will collect the information from the GUI (such as rated power, rated energy, system chemistry, etc) and use it to perform an optimization. After analysis it outputs the results in the results table.

**epBox\_TextChanged** is a simple function – when the energy/power ratio box is changed, the energy box is updated accordingly.

**energyBox\_TextChanged** is a simple function – when the energy box is changed, the energy/power ratio box is updated accordingly.