100 Elwood Davis Road ♦ North Syracuse, NY 13212 ♦ USA

SonnetLab Method Reference

Specialists in High-Frequency Electromagnetic Software
(315) 453-3096 Fax: (315) 451-1694 http://www.sonnetsoftware.com

Interface Method List

This document contains the help notes for the methods available in the SonnetLab toolbox for Matlab (from here on called SonnetLab). Users may retrieve the help information for individual methods by typing 'help SonnetProject.<FunctionName>' into the Matlab command window.

```
SonnetProject   Create a new Sonnet project object
    SonnetProject() Initializes an object to represent a Sonnet project.
    This Sonnet project has the same default settings as what would be generated
    by Sonnet when creating a new geometry project.

    SonnetProject('project.son') Initializes an object to represent a Sonnet
    project. This project object will import all its settings from the
    specified Sonnet project file. The constructor will read the Sonnet
    project information from the file and assign it to the properties of
    the class instantiation.

    See also SonnetProject

ExportHeatFlux   Exports heat flux data
    Project.ExportHeatFlux(...) will call Sonnet and export
    the heat flux data for a region of a layout. This method
    will save and simulate the project first. Current
    calculations will be enabled for the project.

    There are two approaches to calling this method:
    The first approach is the pass the method a
    Sonnet current data request configuration file.

        Example: Project.ExportHeatFlux(aRequestFile);

    The second approach to calling this method involves passing
    arguments that would specify the output settings such that
    the method will essentially build an output configuration
    file. The arguments are the following:

       1) Region - The region must be either a JXYLine object,
                   a JXYRectangle object or []. If the region
                   is [] then the currents for the entire layout
                   will be outputted.
       2) Ports -  The ports should be either a vector of JXYPort
                   objects or a matrix that stores the voltage and
                   phase values for each port. The user only has
                   to define values for ports that have non-zero
                   voltage or phase values. When using a matrix
                   the data must be formatted as follows:
                      [PortNumber, Voltage, Phase;
                       PortNumber, Voltage, Phase; ...]
       3) Frequency -  A vector specifying the desired frequency values.
                       Values should be specified in the same units as the project.
       4) (Optional) X Grid Size - This determines the X direction resolution
                                   of the exported data. The grid size is the
                                   separation between two data points. The first
                                   value in the series is half of the grid size.
                                   Ex: a value of two would provide data at the
                                   points 1,3,5,7... If the grid X size is
                                   unspecified then the cell size from the project
                                   will be utilized.
       5) (Optional) Y Grid Size - This determines the Y direction resolution of
                                   the exported data. If the grid Y size
                                   is unspecified then the cell size from the
                                   project will be utilized.
       6) (Optional) Level -  Specifies what metallization  level(s)
                              should be outputted. The level should be [] if
```

```
                              all levels should be outputted. The level should
                              be a single number (Ex: 4) if only one level
                              should be outputted. If a range of levels
                              should be outputted then the level should be a
                              vector in the form of [startLevel, endLevel].
      7) (Optional) Complex - Should be either true or false. True indicates that
                              current data should be returned as complex numbers.

   If the user would like to specify values for
   parameters they may use the last two arguments.

      8)  ParameterName  -    Should be either a vertical vector of strings
                              (use strvcat) or a cell array of strings.

      9)  ParameterValue -    Should be either a vector or a cell array of values
                              such that the Nth element of ParameterValue is
                              the value for the parameter specified by the Nth
                              element of ParameterName.

   Note: This method is only for geometry projects.
   Note: This method will only work for Sonnet version 13 and later.
           This method will look for Sonnet 13 installations and use the
           one with the latest install date.
   Note: This method will save the project to the hard drive. If
         there hasn't been a filename associated with this project
         an error will be thrown. A filename may be specified using
         the saveAs method (see "help SonnetProject.saveAs")

   See also SonnetProject/viewCurrents,
            SonnetProject/enableCurrentCalculations,
            SonnetProject/disableCurrentCalculations,
            SonnetProject/exportPattern


activateVariableSweepParameter   Activates a variable sweep parameter
   Project.activateVariableSweepParameter(VariableName) will set the
   parameter in use value for the specified parameter in the
   first variable sweep to true.

   Project.activateVariableSweepParameter(VariableName,N) will
   set the parameter in use value for the specified parameter in the
   Nth variable sweep to true.

   See also SonnetProject/deactivateVariableSweepParameter



addAbsEntryFrequencySweep   Adds an 'ABSENTRY' type of sweep to the project
   Project.addAbsEntryFrequencySweep(StartFrequency,EndFrequency) adds a
   'ABSENTRY' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an 'ABSENTRY' type of sweep to the project.
       % the sweep will go from 5 to 10.
       Project.addAbsEntryFrequencySweep(5,10);

 See also SonnetProject/addFrequencySweep



addAbsFmaxFrequencySweep   Adds an 'ABSFMAX' type of sweep to the project
   Project.addAbsFmaxFrequencySweep(StartFrequency,EndFrequency,Maximum) adds a
```

```
   'ABSFMAX' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an 'ABSFMAX' type of sweep to the project.
       % the sweep will go from 5 to 10 looking for a
       % max value of 5.
       Project.addAbsFmaxFrequencySweep(5,10,'S11');

 See also SonnetProject/addFrequencySweep



addAbsFminFrequencySweep   Adds an 'ABSFMIN' type of sweep to the project
   Project.addAbsFminFrequencySweep(StartFrequency,EndFrequency,Minimum) adds a
   'ABSFMIN' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an 'ABSFMIN' type of sweep to the project.
       % the sweep will go from 5 to 10 looking for a
       % min value of 5.
       Project.addAbsFminFrequencySweep(5,10,'S11');

 See also SonnetProject/addFrequencySweep



addAbsFrequencySweep   Adds an 'ABS' type of sweep to the project
   Project.addAbsFrequencySweep(StartFrequency,EndFrequency) adds an
   'ABS' type of frequency sweep to the project.

   This function will change the selected frequency sweep to 'ABS'.

   Example usage:

       % Add an 'ABS' type of sweep to the project.
       % the sweep will go from 5 to 10.
       Project.addAbsFrequencySweep(5,10);

 See also SonnetProject/addFrequencySweep



addAnchoredDimensionParameter   Adds a dimension parameter
   Project.addAnchoredDimensionParameter(...) will add an anchored
   geometry dimension parameter to the project.

   addDimensionParameter eight arguments:
     1) The parameter name (Ex: 'Width')
     2) Handle for first reference polygon or the polygon's ID
     3) The vertex number used for the first reference polygon
     4) Handle for second reference polygon or the polygon's ID
     5) The vertex number used for the second reference polygon
     6) A cell array of any polygons that have points that should
           be altered by this dimension parameter. If there is
           only one polygon to be altered then this parameter
           does not need to be a cell array.
     7) A cell array of vectors that indicate which vertices of
```

```
        the polygon should be altered. If there is
        only one polygon to be altered then this parameter
        does not need to be a cell array.
  8) The direction of movement; this may be 'x','X', or 'XDir'
        for the X direction and 'y','Y', or 'YDir' for the Y
        direction.
  9) (Optional) The equation that should be used.

Note: This method is only for geometry projects.
Note: This method will add dimension parameters to a project.
      To modify the value of a dimension parameter use the
      modifyVariableValue method.

Example usage:

  Example 1:
    % We have a polygon in a project and we want to alter its
    % width with a dimension parameter. This particular polygon
    % has coordinate values of: (10,10),(30,10),(30,40),
    % (10,40),(10,10). The polygon has an ID of seven. The polygon
    % looks like the following diagram with the vertices numbered.
    % We want the polygon to grow/shrink on the right hand side
    % (coordinates 2 and 3) while keeping the left hand
    % (coordinates 1 and 4) constant.
    %
    %                          4-----3
    %                          |     |
    %                          |     |
    %                          |     |
    %                          1-----2
    %
    % To accomplish our goal we can add a dimension parameter
    % to the project. The parameter will be named 'Width' and
    % be attached to the polygon with an ID of seven. The first
    % reference vertex will be the first vertex of the desired
    % polygon and the second vertex value will be the second
    % vertex of the polygon. The two reference points signify
    % a move in the X direction.
    %
    % Now we will add some polygons that have altering points
    % to the point set.  In this case we want to alter the points
    % on the right hand side of the polygon.  The second
    % reference point already corresponds to one of the points;
    % the second point we want to select for movement is the
    % first coordinate of the polygon.

    Project.addAnchoredDimensionParameter('Width',7,3,7,2,7,1,'x');

    % Alternately, the polygon's coordinates could have been selected
    % easier with the polygon methods lowerRightVertex(), lowerLeftVertex(),
    % upperRightVertex(), upperLeftVertex(). These methods will return
    % the index of the coordinate that is at the desired location of
    % the polygon. The polygon coordinate methods are intended for
    % rectangular polygons only. Using the polygon coordinate access
    % methods on non-rectangular polygons could potential yield
    % undesirable results (Example: what is the lower left corner
    % of a spiral? lowerLeftVertex() will return the best value it
    % can but the user should be aware that in that case they may
    % be better off specifying the coordinate manually). In order
    % to use methods such as lowerRightVertex() we will need to
    % obtain a reference to the desired polygon; this can be
    % accomplished using the findPolygonUsingId() method.

    [~, polygon]=Project.findPolygonUsingId(7);
    Project.addAnchoredDimensionParameter('Width',...
            polygon,polygon.lowerLeftVertex(),...
            polygon,polygon.lowerRightVertex(),...
```

```
          polygon,polygon.upperRightVertex(),'x');

Example 2:
  % We have two polygons in a project and we want to alter
  % their separation with a dimension parameter. The left
  % polygon has coordinate values of: (10,10),(30,10),(30,40),
  % (10,40),(10,10). The right polygon has coordinate values
  % of (50,10),(80,10),(80,30),(70,30),(70,40),(50,40). The left polygon
  % has an ID of seven and the right polygon has an ID of eight.
  % The polygon layout looks like the following diagram with the vertices
  % numbered. We want the right polygon to move closer or farther
  % away from the fixed left polygon.
  %
  %             4-----3       6-----5
  %             |     |       |     |__3
  %             |     |       |     4  |
  %             |     |       |        |
  %             1-----2       1--------2
  %
  % To accomplish our goal we will add a dimension parameter. We
  % will call our parameter 'Sep'. The first reference point will
  % be attached to vertex number two of the left polygon (ID of seven)
  % and the second reference point will be attached to vertex
  % number one of the right polygon (ID of eight).
  %
  % Now we will add some polygons that have altering points
  % to the point set.  In this case we want to alter the all
  % the points for the polygon on the right. We may indicate
  % that all the points in the polygon should be altered by
  % not specifying which points in the polygon should be altered.

  Project.addAnchoredDimensionParameter('Sep',7,2,8,1,8,[],'x');

Example 3:
  % We have three polygons in a project and we want to alter the separation
  % between the right two polygons and the left most polygon. The left
  % polygon has coordinate values of: (10,10),(30,10),(30,40),
  % (10,40),(10,10). The middle polygon has coordinate values
  % of (50,10),(80,10),(80,30),(70,30),(70,40),(50,40). The polygon on
  % the right has coordinate values of (90,10),(120,10),(120,30),(110,30),
  % (110,40),(90,40). The left polygon has an ID of seven, the middle
  % polygon has an ID of eight  and the right polygon has an ID of nine.
  % The polygon layout looks like the following diagram with the vertices
  % numbered. We want the middle and right polygons to move closer or farther
  % away from the fixed left polygon.
  %
  %             4-----3       6-----5     6-----5
  %             |     |       |     |__3  |     |__3
  %             |     |       |     4  |  |     4  |
  %             |     |       |        |  |        |
  %             1-----2       1--------2  1--------2
  %
  % To accomplish our goal we will add a dimension parameter. We
  % will call our parameter 'Sep'. The first reference point will
  % be attached to vertex number two of the left polygon (ID of seven)
  % and the second reference point will be attached to vertex
  % number one of the middle polygon (ID of eight).
  %
  % Now we will add some polygons that have altering points
  % to the point set.  In this case we want to alter the all
  % the points for the middle polygon and the right polygon.
  % Because more than one polygon is to be modified we must
  % put the polygons and vertices in cell arrays. Because the
  % entire polygons should be moved the vertices may be specified
  % by the empty set ([]); in this example we will explicitly state
  % the vertices anyway so that the user can see how to indicate
  % individual vertices.
```

```
        aArrayOfPolygons{1}=8;
        aArrayOfPolygons{2}=9;
        aArrayOfPoints{1}=[1 2 3 4 5 6];
        aArrayOfPoints{2}=[1 2 3 4 5 6];
        Project.addAnchoredDimensionParameter('Sep',7,2,8,1,aArrayOfPolygons,aArrayOfPoints,'x');



addAnisotropicDielectricLayer   Add an anisotropic dielectric layer to the project
   Project.addAnisotropicDielectricLayer(...) will add a dielectric
   layer to the top of the project.

   If the layer is anisotropic then it requires the
   following arguments:

       1)  Name of the Dielectric Layer
       2)  Thickness of the layer
       3)  Relative Dielectric Constant
       4)  Relative Magnetic Permeability
       5)  Dielectric Loss Tangent
       6)  Magnetic Loss Tangent
       7)  Dielectric Conductivity
       8)  Relative Dielectric Constant for Z Direction
       9)  Relative Magnetic Permeability for Z Direction
       10) Dielectric Loss Tangent for Z Direction
       11) Magnetic Loss Tangent for Z Direction
       12) Dielectric Conductivity for Z Direction
       13) Number of Z-Partitions (Optional)

   Note: This method is only for geometry projects.

   Example usage:
       % Add a new dielectric layer to the project. The layer
       % is 10 units thick, has a relative dielectric constant
       % of 1, a relative magnetic permeability of 1,
       % a dielectric loss tangent of 0, a magnetic loss
       % tangent of 0, a dielectric conductivity of 0.
       % The Z direction has a relative dielectric constant
       % of 1, a dielectric loss tangent of 1, a magnetic
       % loss tangent of 0, and an dielectric conductivity of 0.
       Project.addAnisotropicDielectricLayer('newLayer',10,1,1,0,0,0,1,1,0,0,0);

 See also SonnetProject/addDielectricLayer



addCapacitorComponent   Add a capacitor component
   Project.addCapacitorComponent(...) adds an ideal capacitor
   component to a geometry project.

   addCapacitorComponent takes the following arguments:
     1) The component name (Ex: 'C1')
     2) The capacitor value (Ex: 50)
     3) Level number
     4) A nx2 matrix of the component port locations.
           The first row should be the first port's X value, then its Y value
           The second row should be the second port's X value, then its Y value
              etc.
     5) (Optional) The terminal width
           This value should be either
               - "Feed" to use the feedline width (Default)
               - "Cell" for one cell width
               - A number which represents a custom width

   Note: This method is only for geometry projects.
   Note: This method will add components to a project.
```

```
          To modify the value of a component use the
          modifyComponentValue method.

   Example usage:
       Project.addCapacitorComponent('C1',50,0,[104.5 156; 104.5 189])
       Project.addCapacitorComponent('C2',50,0,[104.5 156; 104.5 189],5)
       Project.addCapacitorComponent('C3',50,0,[104.5 156; 104.5 189],'Feed')
       Project.addCapacitorComponent('C4',50,0,[104.5 156; 104.5 189],'1Cell')



addCapacitorElement   Creates a capacitor element
   Project.addCapacitorElement(Node1,Node2,Capacitance) will add
   an capacitor element to the circuit between Node1 and Node2 with
   the specified capacitance. If the second node of the capacitor
   should not be attached to any node then Node2 should be [].

   Project.addCapacitorElement(Node1,Node2,Capacitance,Network) will add
   an capacitor element to the specified network of the circuit between
   Node1 and Node2 with the specified capacitance. If the second node
   of the capacitor should not be attached to any node then Node2 should
   be []. The network selection may be the network's index or the
   network's name.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a capacitor element to the first network
       % in the project. The capacitor is connected
       % from node 1 to 2 with capacitance of 50
       Project.addCapacitorElement(1,2,50);

       % Add a capacitor element to the second network
       % in the project. The capacitor is connected
       % from node 1 to 2 with capacitance of 50
       Project.addCapacitorElement(1,2,50,2);

   See also SonnetProject/addResistorElement,
            SonnetProject/addInductorElement,
            SonnetProject/addTransmissionLineElement,
            SonnetProject/addPhysicalTransmissionLineElement,
            SonnetProject/addDataResponseFileElement,
            SonnetProject/addProjectFileElement,
            SonnetProject/addNetworkElement



addCoCalibratedGroup   Add a co-calibrated port group
   Project.addCoCalibratedGroup(name,GroundReference,TerminalWidthType) will
   add a co-calibration group to the array of co-calibration groups.

   Note: This method is only for geometry projects.

   Example usage:
       Project.addCoCalibratedGroup('A','B','FEED');

     See also SonnetProject/addPort, SonnetProject/addPortCocalibrated



addComment   Adds a comment to a Sonnet project
   Project.addComment(theString) adds passed text
   as a new comment stored in the project file.

   Note: Comments are stored in the project file but
         are not displayed in the Sonnet project editor.
```

```
addDataFileComponent   Add a data file component
   Project.addDataFileComponent(...) adds a data
   file component to a geometry project.

   addDataFileComponent takes the following arguments:
     1) The component name (Ex: 'R1')
     2) The data file name (Ex: 'Project.s2p')
     3) Level number
     4) A nx2 matrix of the component port locations.
           The first row should be the first port's X value, then its Y value
           The second row should be the second port's X value, then its Y value
              etc.
     5) (Optional) The terminal width
           This value should be either
               - "Feed" to use the feedline width (Default)
               - "Cell" for one cell width
               - A number which represents a custom width

   Note: This method is only for geometry projects.
   Note: This method will add components to a project.
         To modify the value of a component use the
         modifyComponentValue method.

   Example usage:
       Project.addDataFileComponent('DF1','Project.s2p',0,[104.5 156; 104.5 189])
       Project.addDataFileComponent('DF2','Project.s2p',0,[104.5 156; 104.5 189],5)
       Project.addDataFileComponent('DF3','Project.s2p',0,[104.5 156; 104.5 189],'Feed')
       Project.addDataFileComponent('DF4','Project.s2p',0,[104.5 156; 104.5 189],'1Cell')



addDataResponseFileElement   Creates a data response file element
   Project.addDataResponseFileElement(Filename,PortNodes) will
   add a SnP file to the circuit connected to the ports
   specified by PortNodes.

   Project.addDataResponseFileElement(Filename,PortNodes,Network) will
   add a SnP file to the circuit connected to the ports specified
   by PortNodes. The network selection may be the network's index
   or the network's name.

   Project.addDataResponseFileElement(Filename,PortNodes,Network,GroundNode)
   will add a SnP file to the circuit connected to the ports specified
   by PortNodes and grounded at the specified ground node number. The
   network selection may be the network's index or the network's name.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a data response file element to the first network of the project
       Project.addDataResponseFileElement('data.s2p',[1,2]);

       % Add a data response file element to the second network of the project
       Project.addDataResponseFileElement('data.s2p',[1,2],2);

       % Add a data response file element to the second network of the project
       % and has its ground reference node connected to node 1.
       Project.addDataResponseFileElement('data.s2p',[1,2],2,1);

   See also SonnetProject/addResistorElement,
            SonnetProject/addInductorElement,
            SonnetProject/addCapacitorElement,
            SonnetProject/addTransmissionLineElement,
```

```
            SonnetProject/addPhysicalTransmissionLineElement,
            SonnetProject/addProjectFileElement,
            SonnetProject/addNetworkElement



addDcFrequencySweep   Adds an 'DC' type of sweep to the project
   Project.addDcFrequencySweep('AUTO') adds an automatic 'DC'
   type of frequency sweep to the project.

   Project.addDcFrequencySweep('MAN',Frequency) adds an manual
   'DC' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an automatic DC frequency sweep to the project
       Project.addDcFrequencySweep('AUTO');

       % Add a manual DC frequency sweep to the project with frequency 5
       Project.addDcFrequencySweep('MAN',5);

 See also SonnetProject/addFrequencySweep



addDielectricBrick   Add a dielectric brick polygon to the polygon array
   Project.addDielectricBrick(...) will add a polygon
   to the array of polygons.

   addDielectricBrick requires these arguments:
     1)  metallization Level Index (The level the polygon is on)
     2)  The material used for the polygon. This may either
          be a the index for the brick material type in the
          array of brick types, Or the name of the material
          (Ex: 'Air'). Air is not in the array of isotropic
          or anisotropic materials but can be selected by
          either passing 0 or 'Air'.
     3)  Minimum subsection size in X direction
     4)  Minimum subsection size in Y direction
     5)  Maximum subsection size in X direction
     6)  Maximum subsection size in Y direction
     7)  The Maximum Length for The Conformal Mesh Subsection
     8)  Edge mesh setting. Y indicates edge meshing is on for this
           polygon. N indicates edge meshing is off.
     9)  A matrix for the X coordinate values
     10) A matrix for the Y coordinate values

   Note: Many users will prefer to use the 'addDielectricBrickEasy' method.
   Note: This method is only for geometry projects.

   Example usage:
       % Metal at level 0, material type 0 (Air),
       % X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
       Project.addDielectricBrick(0,0,0,0,50,100,0,'Y',x,y);

       % Metal at level 0, material type Brick1,
       % X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
```

```
        Project.addDielectricBrick(0,'Brick1',0,0,50,100,0,'Y',x,y);

    See also SonnetProject/addDielectricBrickEasy



 addDielectricBrickEasy   Add a dielectric brick polygon to the polygon array
    Polygon=Project.addDielectricBrickEasy(...) will add a dielectric brick
    to the array of polygons. A reference to the polygon
    is returned.

    addDielectricBrickEasy requires these arguments:
      1) metallization  Level Index (The level the polygon is on)
      2) A column vector for the X coordinate values
      3) A column vector for the Y coordinate values
      4) (Optional) The material used for the polygon. This may
          either be a the index for the brick material type in
          the array of brick types, or the name of the material
          (Ex: 'Air'). If this value is not specified the
          function will use 'Air'.

    Note: This method is only for geometry projects.

    Example usage:
        % Build a brick on layer zero of type 'Air'
        Project.addDielectricBrickEasy(0,[5,10,10,5,5],[10,10,20,20,10]);

        % Build a brick on layer zero of type 'Brick1'
        Project.addDielectricBrickEasy(0,[5,10,10,5,5],[10,10,20,20,10],'Brick1');

    See also SonnetProject/addDielectricBrick



 addDielectricLayer   Add a dielectric layer to the project
    Project.addDielectricLayer(...) will add a dielectric layer
    to the top of the stackup (the end of the array of dielectric
    layers).

    There are two ways to use addDielectricLayer. The user
    may define a layer using a set of custom options or
    the user may define a using a predefined property set
    from the Sonnet library.

    Users may use addDielectricLayer to add a custom dielectric
    layer to the project using the following parameters:
        1)  Name of the Dielectric Layer
        2)  Thickness of the layer
        3)  Relative Dielectric Constant
        4)  Relative Magnetic Permeability
        5)  Dielectric Loss Tangent
        6)  Magnetic Loss Tangent
        7)  Dielectric Conductivity
        8)  Number of Z-Partitions (Optional)

    Users may add a layer based on an entry from the Sonnet
    library by using the following parameters:
        1) The name of the material (Ex: "Rogers RT6006")
        2) Thickness of the layer

    If no dielectric layer exists in the SonnetLibrary
    with the specified name then an error will be thrown.

    Note: This method is only for geometry projects.

    Example usage:
        % Add a new dielectric layer to the project. The layer
```

```
       % is 10 units thick, has a relative dielectric constant
       % of 1, a relative magnetic permeability of 1,
       % a dielectric loss tangent of 0, a magnetic loss
       % tangent of 0, an dielectric conductivity of 0.
       Project.addDielectricLayer('newLayer',10,1,1,0,0,0);

       % This layer is the same as the one above but
       % it specifies that there are 2 Z-partitions.
       Project.addDielectricLayer('newLayer2',10,1,1,0,0,0,2);

       % This layer uses Rogers RT6006
       Project.addDielectricLayer('Rogers RT6006',50);
```

See also SonnetProject/addAnisotropicDielectricLayer

addDimensionLabel

Adds a dimension label

Project.addDimensionLabel(...) will add a dimension label to the project.

addDimensionLabel eight arguments:

1) Handle for first reference polygon or the polygon's ID
2) The vertex number used for the first reference polygon
3) Handle for second reference polygon or the polygon's ID
4) The vertex number used for the second reference polygon
5) The direction of movement; this may be 'x','X', or 'XDir' for the X direction and 'y','Y', or 'YDir' for the Y direction.

Note: This method is only for geometry projects.

Example usage:

Example 1:

```
       % We have a polygon in a project and we want to mark its
       % width with a dimension label. This particular polygon
       % has coordinate values of: (10,10),(30,10),(30,40),
       % (10,40),(10,10). The polygon has an ID of seven. The polygon
       % looks like the following diagram with the vertices numbered.
       % We want to place a dimension label between
       % coordinates 1 and 2.
       %
       %                          4-----3
       %                          |     |
       %                          |     |
       %                          |     |
       %                          1-----2
       %
       % The two reference polygon inputs will be the same
       % polygon. The reference polygon can be specified with
       % its debug ID which is seven. The label can be added
       % with the following command:

       Project.addDimensionLabel(7,1,7,2,'x');

       % Alternately, the polygon's coordinates could have been selected
       % easier with the polygon methods lowerRightVertex(), lowerLeftVertex(),
       % upperRightVertex(), upperLeftVertex(). These methods will return
       % the index of the coordinate that is at the desired location of
       % the polygon. The polygon coordinate methods are intended for
       % rectangular polygons only. Using the polygon coordinate access
       % methods on non-rectangular polygons could potential yield
       % undesirable results (Example: what is the lower left corner
       % of a spiral? lowerLeftVertex() will return the best value it
       % can but the user should be aware that in that case they may
       % be better off specifying the coordinate manually). In order
```

```
        % to use methods such as lowerRightVertex() we will need to
        % obtain a reference to the desired polygon; this can be
        % accomplished using the findPolygonUsingId() method.

        [~, polygon]=Project.findPolygonUsingId(7);
        Project.addDimensionLabel(polygon,polygon.lowerLeftVertex(),...
                polygon,polygon.lowerRightVertex(),'x');
```

See also SonnetProject/addAnchoredDimensionParameter,
SonnetProject/addSymmetricDimensionParameter,

addEdgeVia Add a new edge via

Project.addEdgeVia(Polygon,EdgeNumber,Level) will add an Edge Via to a polygon in the project. Polygon may be either a reference to a polygon object or the polygon's ID. The via is placed on the polygon edge between the specified number and the next number. For example, if vertex 3 is specified, the via extends from vertex 3 to vertex 4 on the polygon. Level should be either the index of the metallization level the via should be attached to or 'GND' or 'TOP'.

Note: This method is only for geometry projects.

Example usage:

```
        % Add an edge via to the polygon with
        % debug ID 8 at vertex number 1. The via
        % will be connected to layer 0.
        Project.addEdgeVia(8,1,0);

        % Add an edge via to the polygon with
        % debug ID 8 at vertex number 2. The via
        % will be connected to 'GND'.
        Project.addEdgeVia(8,2,'GND');
```

addEsweepFrequencySweep Adds an 'ESWEEP' type of sweep to the project

Project.addEsweepFrequencySweep(StartFrequency,EndFrequency,NumberOfPoints) adds a 'ESWEEP' type of frequency sweep to the project.

This sweep is part of a combination frequency sweep. This function will change the selected frequency sweep to frequency sweep combination.

Example usage:

```
        % Add an 'ESWEEP' type of sweep to the project.
        % the sweep will go from 5 to 10 with 5 points.
        Project.addEsweepFrequencySweep(5,10,5);
```

See also SonnetProject/addFrequencySweep

addFileOutput Create a new output file

Project.addFileOutput(...) will add another output file to the project. This method takes the following arguments:

1) A string to represent the File Type as follows:

File Type	Entry Definition
TS	Touchstone
DATA_BANK	Databank
SC	SCompact
CSV	Spreadsheet

```
        CADENCE         Cadence
        MDIF            MDIF (S2P)
        EBMDIF          MDIF (ebridge)

   2) The Network Name to be exported (only applies to Netlist).  If you want
      the output of all networks then have this argument be the empty string
      ( '' ). This parameter can be completely ignored in most cases.

   3) Whether or not to embed. This field is "D" for de-embedded data
        or "ND" for non-de-embedded data.

   4) This field is "Y" to include the ABS adaptive data or "N" to
        include only the discrete data.

   5) The filename consists of a basename and extension. If the basename
        of the project file is used, the variable "$BASENAME" may be
        substituted in the filename. For example, in the project file
        steps.son if an output file steps.s2p is entered, the filename
        would appears as "$BASENAME.s2p" in the fileout block. The user may
        enter any filename they wish and is not restricted in their
        use of extensions.

   6) This field is "NC" for no comments or "IC" to include comments.

   7) This field is 'Y' if the output is high precision and 'N' if not.

   8) This field is "S" for S-Parameters, "Y" for Y-Parameters,
        and "Z" for Z-Parameters. This value is 'SPECTRE'
        for NCLINE (RLGC) file outputs. If the output is
        NCLINE then do not include any of the below
        arguments.

   9) The form for the Parameter has the following entry possibilities
         MA  -  Mag-Angle
         DB  -  DB-Angle
         RI  -  Real-Imaginary

   10) The PortType should be one of the following
         R       If all ports in the circuit use real impedance
                    with the same resistance and all other values 0

         Z       If all ports in the circuit use complex impedance
                    with the same resistance and all other values 0

         TERM    If a port or ports in the circuit have a non-zero value
                    for either the Resistance or Reactance

         FTERM   If a port or ports in the circuit have a non-zero value
                    for the Resistance or Reactance and either
                    the inductance or capacitance

If the port type was resistor
   11) One or more Resistance values stored as a matrix

If the port type was complex impedance
   11) One or more Resistance values stored as a matrix
   12) One or more ImaginaryResistance

If the port type was TERM
   11) One or more Resistance values stored as a matrix
   12) One or more Reactance values stored as a matrix

If the port type was FTERM
   11) One or more Resistance values stored as a matrix
   12) One or more Reactance values stored as a matrix
   13) One or more Inductance values stored as a matrix
   14) One or more Capacitance values stored as a matrix
```

```
   Example usage:

       % Add a new touchstone file output to the project
       % the name out the outputted file will be the name
       % name of the project ('BASENAME' gets replaced
       % with the project name automatically)
       Project.addFileOutput('TS','D','Y','$BASENAME.s1p','IC','N','S','MA','R',20);

   See also SonnetProject/addFileOutputForNetlist,
            SonnetProject/addFileOutputForGeometry



addFileOutputForGeometry   Create a new output file
   Project.addFileOutputForGeometry(...) will add another output file to the
   project. This method was not meant to be called
   directly; please use addFileOutput instead to make
   sure the project is a geometry project.

   Type 'help SonnetProject.addFileOutput' for arguments and more information.

   See also SonnetProject/addFileOutput, SonnetProject/addFileOutputForNetlist



addFileOutputForNetlist   Create a new output file
 Project.addFileOutputForNetlist(...) will add another output file to the
 project. This method was not meant to be called
 directly; please use addFileOutput instead to make
 sure the project is a netlist project.

   Type 'help SonnetProject.addFileOutput' for arguments and more information.

   See also SonnetProject/addFileOutput, SonnetProject/addFileOutputForGeometry



addFrequencySweep   Adds a frequency sweep to the project
   Project.addFrequencySweep(SweepName,...) adds a frequency sweep
   to the project. addFrequencySweep requires a string specifying the type
   of frequency sweep to be added to the project and
   all of the arguments necessary in order to construct
   the sweep.

   Types and arguments are as follows:

       SWEEP       StartFrequency,EndFrequency,StepFrequency
       ABS         StartFrequency,EndFrequency
       ABSENTRY    StartFrequency,EndFrequency
       ABSFMAX     StartFrequency,EndFrequency,Maximum
       ABSFMIN     StartFrequency,EndFrequency,Minimum
       DC          Mode*,Frequency**
       ESWEEP      StartFrequency,EndFrequency,AnalysisFrequencies
       LSWEEP      StartFrequency,EndFrequency,AnalysisFrequencies
       SIMPLE      StartFrequency,EndFrequency,StepFrequency
       STEP        StepFrequency

       *  For a DC sweep:  mode is either 'AUTO' for automatic or 'MAN' for manual.
       ** For a DC sweep:  when mode is 'AUTO' the frequency does not need to
                           be supplied. The frequency is required when the DC
                           mode is manual.

   When a frequency sweep is added to the project the selected
   frequency sweep to be used for analysis will be automatically
   changed such that the newly created sweep will be the selected
   frequency sweep.
```

```
   Example usage:

       % Add an ABS sweep to the project. The new sweep will
       % have the frequency range from 5 to 10 (units are
       % specified in the dimension block)
       Project.addFrequencySweep('ABS',5,10);

       % Add an automatic DC frequency sweep to the project
       Project.addFrequencySweep('DC','AUTO');

       % Add a manual DC frequency sweep to the project with frequency 5
       Project.addFrequencySweep('DC','MAN',5);

 See also SonnetProject/addSweepFrequencySweep,
          SonnetProject/addAbsFrequencySweep,
          SonnetProject/addAbsEntryFrequencySweep,
          SonnetProject/addAbsFmaxFrequencySweep,
          SonnetProject/addAbsFminFrequencySweep,
          SonnetProject/addDcFrequencySweep,
          SonnetProject/addEsweepFrequencySweep,
          SonnetProject/addLsweepFrequencySweep,
          SonnetProject/addSimpleFrequencySweep,
          SonnetProject/addStepFrequencySweep



addInductorComponent   Add a inductor component
   Project.addInductorComponent(...) adds an ideal inductor
   component to a geometry project.

   addInductorComponent takes the following arguments:
     1) The component name (Ex: 'L1')
     2) The inductor value (Ex: 50)
     3) Level number
     4) A nx2 matrix of the component port locations.
           The first row should be the first port's X value, then its Y value
           The second row should be the second port's X value, then its Y value
              etc.
     5) (Optional) The terminal width
           This value should be either
               - "Feed" to use the feedline width (Default)
               - "Cell" for one cell width
               - A number which represents a custom width

   Note: This method is only for geometry projects.
   Note: This method will add components to a project.
         To modify the value of a component use the
         modifyComponentValue method.

   Example usage:
       Project.addInductorComponent('L1',50,0,[104.5 156; 104.5 189])
       Project.addInductorComponent('L2',50,0,[104.5 156; 104.5 189],5)
       Project.addInductorComponent('L3',50,0,[104.5 156; 104.5 189],'Feed')
       Project.addInductorComponent('L4',50,0,[104.5 156; 104.5 189],'1Cell')



addInductorElement   Creates a inductor element
   Project.addInductorElement(Node1,Node2,Inductance) will add
   an inductor element to the circuit between Node1 and Node2 with
   the specified inductance. If the second node of the inductor
   should not be attached to any node then Node2 should be [].

   Project.addInductorElement(Node1,Node2,Inductance,Network) will add
   an inductor element to the specified network of the circuit between
   Node1 and Node2 with the specified inductance. If the second node
```

```
   of the inductor should not be attached to any node then Node2 should
   be []. The network selection may be the network's index or the
   network's name.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a inductor element to the first network
       % in the project. The inductor is connected
       % from node 1 to 2 with inductance of 50
       Project.addInductorElement(1,2,50);

       % Add a inductor element to the second network
       % in the project. The inductor is connected
       % from node 1 to 2 with inductance of 50
       Project.addInductorElement(1,2,50,2);

   See also SonnetProject/addResistorElement,
            SonnetProject/addCapacitorElement,
            SonnetProject/addTransmissionLineElement,
            SonnetProject/addPhysicalTransmissionLineElement,
            SonnetProject/addDataResponseFileElement,
            SonnetProject/addProjectFileElement,
            SonnetProject/addNetworkElement



addLsweepFrequencySweep   Adds an 'LSWEEP' type of sweep to the project
   Project.addLsweepFrequencySweep(StartFrequency,EndFrequency,NumberOfPoints)
   adds a 'LSWEEP' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an 'LSWEEP' type of sweep to the project.
       % the sweep will go from 5 to 10 with 5 points.
       Project.addLsweepFrequencySweep(5,10,5);

 See also SonnetProject/addFrequencySweep



addMetalPolygon   Add a metal polygon to the polygon array
   Project.addMetalPolygon(...) will add an polygon
   to the array of polygons.

   addMetalPolygon requires these arguments:
      1)  metallization  Level Index ( The level the polygon is on)
      2)  The type of metal used for the polygon. This may either
           be a the index for the metal type in the array of
           metal types, or the name of the metal type
           (Ex: 'Copper'). Lossless metal is not in the array
           of metals but can be selected by either passing 0
           or 'Lossless'.
      3)  A string to identify the fill type used for the polygon.
           N indicates staircase fill, T indicates diagonal
           fill and V indicates conformal mesh.
      4)  Minimum subsection size in X direction
      5)  Minimum subsection size in Y direction
      6)  Maximum subsection size in X direction
      7)  Maximum subsection size in Y direction
      8)  The Maximum Length for The Conformal Mesh Subsection
      9)  Edge mesh setting. Y indicates edge meshing is on for this
```

```
          polygon. N indicates edge meshing is off.
      10) A column vector for the X coordinate values
      11) A column vector for the Y coordinate values

   Note: Many users will prefer to use the 'addMetalPolygonEasy' method.
   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:
       % metal at level 0, metal type -1 (lossless),
       % staircase fill, X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
       Project.addMetalPolygon(0,0,'N',0,0,50,100,0,'Y',x,y);

       % metal at level 0, metal type 'ThinCopper',
       % staircase fill, X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
       Project.addMetalPolygon(0,'ThinCopper','N',0,0,50,100,0,'Y',x,y);

   See also SonnetProject/addMetalPolygonEasy



addMetalPolygonEasy   Add a metal polygon to the polygon array
   Polygon=Project.addMetalPolygonEasy(...) will add an polygon
   to the array of polygons. A reference to the polygon
   is returned.

   addMetalPolygonEasy requires these arguments:
      1) metallization  Level Index (The level the polygon is on)
      2) A column vector for the X coordinate values
      3) A column vector for the Y coordinate values
      4) (Optional) The type of metal used for the polygon.
           This may either be a the index for the metal
           type in the array of metal types, or the name
           of the metal type (Ex: 'Copper'). If this value
           is not specified then lossless metal will be used.

   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:
       % Build a lossless metal polygon on layer zero
       Project.addMetalPolygonEasy(0,[5,10,10,5,5],[10,10,20,20,10]);

       % Build a copper metal polygon on layer zero (the Copper
       % metal type must be defined in the project)
       Project.addDielectricBrickEasy(0,[5,10,10,5,5],[10,10,20,20,10],'Copper');

   See also SonnetProject/addMetalPolygon



addNCoupledLineOutput   Create a new N-Coupled Line Model
   Project.addNCoupledLineOutput(...) will add a N-Coupled
   line model output file to the project.

   Arguments are:
      1) Whether or not to use embedded data. This field is "D"
```

```
          for de-embedded data or "ND" for non-de-embedded data.
       2) This field is "Y" to include the ABS adaptive data or "N" to
          include only the discrete data.
       3) The filename consists of a basename and extension. If the basename
          of the project file is used, the variable "BASENAME" may be
          substituted in the filename. For example, in the project file
          steps.son if an output file steps.s2p is entered, the filename
          would appears as "$BASENAME.dat" in the fileout block. The user may
          enter any filename they wish and is not restricted in their
          use of extensions.
       4) This field is 'Y' if the output is high precision and 'N' if not.
       5) (Optional) When used with a netlist project this
          argument allows users to output data for only a
          specified network by name.

   Example:
       Project.addNCoupledLineOutput('D','Y','$BASENAME.dat','Y');
       Project.addNCoupledLineOutput('D','Y','$BASENAME.dat','Y','Network1');

   See also SonnetProject/addFileOutput



addNetworkElement   Creates a network element
   Project.addNetworkElement(...) will add an network element to the circuit

       addNetworkElement takes the following parameters:

              1) The name for the new network
              2) The vector of port numbers

       And then also include one of the following:

        * If you want to define a single real impedance for all the ports then:
              3) the impedance

        * If you want to define a single non-real impedance for all the ports then:
              3) the real component of the impedance
              4) the imaginary component of the impedance

        * If you want to define different resistances and reactances for each port
          then pass the following for an N dimensional network:
              3) An  N x 2  matrix with the first column being the
              resistance of the port and the second number
              being the reactance of the port. Each row in
              the matrix should correspond to a single port
              and be specified in the same order as was
              specified in the second argument which was
              an vector of port numbers.

        * If a port or ports in the circuit have non-zero values for either the
          inductance or capacitance then pass the following:
              3) An  N x 4  matrix with the first column being the
              resistance of the port, the second number
              being the reactance of the port, the third column
              is for the inductance of the port and the fourth
              is for the capacitance of the port. Each row in
              the matrix should correspond to a single port
              and be specified in the same order as was
              specified in the second argument which was
              an vector of port numbers.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a new network to the project. All ports will
```

```
        % have a real impedance of 50.
        Project.addNetworkElement('NetName1',[1 2 3 4],50);

        % Add a new network to the project. All ports will
        % have a real impedance of 50 and an imaginary component
        % of 50.
        Project.addNetworkElement('NetName2',[1 2 3 4],50,50);

        % Add a new network to the project. All ports will
        % have a differing resistances and reactances.
        Project.addNetworkElement('NetName3',[1 2 3 4],[50 50; 100 100]);

        % Add a new network to the project. All ports will
        % have a differing resistances, reactances,
        % inductances, and capacitances.
        Project.addNetworkElement('NetName4',[1 2 3 4],[50 50 10 10; 100 100 10 10]);

    See also SonnetProject/addResistorElement,
             SonnetProject/addInductorElement,
             SonnetProject/addCapacitorElement,
             SonnetProject/addTransmissionLineElement,
             SonnetProject/addPhysicalTransmissionLineElement,
             SonnetProject/addDataResponseFileElement,
             SonnetProject/addProjectFileElement



addOptimizationParameter   Create a new optimization parameter
    Project.addOptimizationParameter(...) adds a new optimization
    parameter to the optimization block. Optimization
    parameters define how the optimization variables get modified.

    addOptimizationParameter requires the following inputs:
        1)  A frequency sweep object. The frequency sweep
            cannot be SonnetFrequencyAbs or SonnetFrequencySimple.
            but SonnetFrequencyAbsEntry and SonnetFrequencySweep
            can be used instead and correspond to the same sweeps.
        2)  The response type (Ex: 'DB[S11]')
        3)  The relation type ('>', '<', '=')
        4)  The type for the target response ('VALUE','NET','FILE').
            This is what the response will be compared to.
        5)  The target value. For targets of type 'VALUE' this
            will store the response value we would like
            to obtain from optimization. For 'NET'
            this argument stores the name of the network
            to compare to. For type 'FILE' this stores
            the name of the file that should be used.
        6)  If the target type is 'FILE' or 'NET' then
            the response type for the target value is
            required. If the type is 'VALUE' then this
            should be the empty string ('');
        7)  The weight for this optimization parameter. This
            value is often 1.

    Example usage:

        % Make an empty frequency sweep
        theSweep=SonnetFrequencyAbsEntry();

        % Assign values to the frequency sweep properties
        theSweep.StartFreqValue=1;
        theSweep.EndFreqValue=5;

        % Add the optimization parameter to the project
        Project.addOptimizationParameter(theSweep,'DB[S11]','=','VALUE',-20,1,1)
```

```
addOption   Adds values to option string
   addOption(str) will add the passed option string
     to the defined set of project options.



addParallelSubsection   Adds a parallel subsection
   Project.addParallelSubsection(Side,Length) will add a
   specified length Parallel Subsection to the
   project. Side may be 'LEFT', 'RIGHT', 'TOP',
   or 'BOTTOM'.

   Note: This method is only for geometry projects.

   Example usage:

       % Add a parallel subsection to the 'TOP' of length 12
       Project.addParallelSubsection('TOP',12);



addPhysicalTransmissionLineElement   Creates a physical transmission line element
   Project.addPhysicalTransmissionLineElement(...) will add an physical
   transmission line element to the circuit.

   addPhysicalTransmissionLineElement takes the following parameters:
     1) The first node number to which the line is connected to
     2) The second node number to which the line is connected to
          (If the element is not to be connected to another node
           then pass [] as for the value for the second node number)
     3) The value for the impedance of the line
     4) The value for the length of the line
     5) The value for the frequency of the line
     6) The value for the eeff of the line
     7) The value for the attenuation of the line
     8) (Optional) The index of the network in the array of networks
        If this is not specified the element will be added to the
        first network.
     9) (Optional) The node number that acts as ground for the line.
        In order to specify a ground node the user must specify
        the network (argument number 8 must be included in order to
        specify argument number 9)

   Note: This method is only for netlist projects.

   Example usage:

       % Add a physical transmission line element to the first
       % network of the project. The transmission line will be
       % connected from node 1 to 2 with an impedance of 100,
       % a length of 1000, a frequency of 10, an eeff of 1,
       % and an attenuation of 10.
       Project.addPhysicalTransmissionLineElement(1,2,100,1000,10,1,10);

       % Add a physical transmission line element to the second
       % network of the project. The transmission line will be
       % connected from node 1 to 2 with an impedance of 100,
       % a length of 1000, a frequency of 10, an eeff of 1,
       % and an attenuation of 10.
       Project.addPhysicalTransmissionLineElement(1,2,100,1000,10,1,10,2);

       % Add a physical transmission line element to the second
       % network of the project. The transmission line will be
       % connected from node 1 to 2 with an impedance of 100,
       % a length of 1000, a frequency of 10, an eeff of 1,
       % and an attenuation of 10. The transmission line will
```

```
        % grounded at port 1.
        Project.addPhysicalTransmissionLineElement(1,2,100,1000,10,1,10,2,1);

   See also SonnetProject/addResistorElement,
            SonnetProject/addInductorElement,
            SonnetProject/addCapacitorElement,
            SonnetProject/addTransmissionLineElement,
            SonnetProject/addDataResponseFileElement,
            SonnetProject/addProjectFileElement,
            SonnetProject/addNetworkElement



addPiModel   Create a new Pi Model output file
   Project.addPiModel(...) will add a pi model output file
   to the project.

   Arguments are:
      1) The format for the export. Should be either 'PSPICE'
           or 'SPECTRE'
      2) Whether or not to use embedded data. This field is "D"
           for de-embedded data or "ND" for non-de-embedded data.
      3) This field is "Y" to include the ABS adaptive data or "N" to
           include only the discrete data.
      4) The filename consists of a basename and extension. If the basename
           of the project file is used, the variable "$BASENAME" may be
           substituted in the filename. For example, in the project file
           steps.son if an output file steps.s2p is entered, the filename
           would appears as "$BASENAME.s2p" in the fileout block. The user may
           enter any filename they wish and is not restricted in their
           use of extensions.
      5) This field is "NC" for no comments or "IC" to include comments.
      6) This field is 'Y' if the output is high precision and 'N' if not.
      7) This is a floating point number for the percentage used to
           determine the intervals between the two frequencies used to
           determine each SPICE model.
      8) This is a floating point number for the percentage used to determine the
          intervals between the two frequencies used to determine each SPICE model
      9) This is a floating point number for the maximum allowed resistance
     10) This is a floating point number for the minimum allowed capacitance
     11) This is a floating point number for the maximum allowed inductance
     12) This is a floating point number for the minimum allowed mutual inductance
     13) This is a floating point number for the resistor to go in series with
           all lossless inductors

   See also SonnetProject/addFileOutput



addPolygon   Adds a polygon object to the project
   Project.addPolygon(Polygon) will add the passed
   polygon to the end of the array of polygons.

   Note: This method is only for geometry projects.

   See also SonnetProject/viewCurrents,
            SonnetProject/enableCurrentCalculations



addPort   Add a port to the project
   Port=Project.addPort(...) will add a port to the project.
   This method is only for geometry projects. A reference to
   the new port is returned.

   addPort requires a type as
   the first argument which should
```

```
be one of the following:

    STD   -   Standard Port
    AGND  -   Auto Grounded Port
    CUP   -   Co-Calibrated Port

Then you will need to supply the necessary
arguments for each as follows:

STD - Standard Port
  1) The Polygon to which the port is attached.
        This can be replaced by the polygon's
        debug ID value.
  2) The Vertex to which the polygon is attached
  3) The Resistance for the port
  4) The Reactance for the port
  5) The Inductance for the port
  6) The Capacitance for the port
  7) The Port Number (Optional)

AGND - Auto Grounded Port
  1) The Polygon to which the port is attached.
        This can be replaced by the polygon's
        debug ID value.
  2) The Vertex to which the polygon is attached
  3)  The Resistance for the port
  4)  The Reactance for the port
  5)  The Inductance for the port
  6)  The capacitance for the port
  7)  A character string which identifies a
       reference plane for the autogrounded port.
       This value is FIX for a reference
       plane and NONE for a calibration length.
  8)  A floating point number which provides the
       length of the reference plane when the type
       is FIX and provides the calibration length
       when the type is NONE.
  9) The Port Number(Optional)

CUP - Co-calibrated Port
  1) The Polygon to which the port is attached.
        This can be replaced by the polygon's
        debug ID value.
  2) The Name of the group to which it belongs
  2) The Vertex to which the polygon is attached
  4) The Resistance for the port
  5) The Reactance for the port
  6) The Inductance for the port
  7) The capacitance for the port
  8) The Port Number (Optional)

Note: This method is only for geometry projects.

Example usage:
    % Add a standard port
    Project.addPort('STD',11,1,75,0,0,0);

    % Add an autogrounded port
    Project.addPort('AGND',11,1,50,0,0,0,'FIX',10);

    % Add an co-calibrated port
    PortReference=Project.addPort('CUP',11,'A',1,75,0,0,0);

See also SonnetProject/addPortToPolygon, SonnetProject/addPortCocalibrated,
         SonnetProject/addPortAtLocation, SonnetProject/addPortStandard,
         SonnetProject/addPortAutoGrounded
```

```
addPortAtLocation   Add a port to the project
   Port=Project.addPortAtLocation(X,Y) will add an standard port
   to the project by specifying an X and Y coordinate.
   The function will find the closest polygon edge and
   place the port there. A reference to the new port is returned.

   Port=Project.addPortAtLocation(X,Y,Level) will add an standard port
   to the project by specifying an X and Y coordinate.
   The function will find the closest polygon edge and
   place the port there. Only polygons on the specified
   level will be checked. A reference to the new port is returned.

   Note: This method is only for geometry projects.
   Note: If the distance between the closest edge and the port
         location is more than 5% of the average of the length
         and width of the box then the port will not be placed
         and an error will be thrown.

   Example usage:
       % Add a standard port
       Port=Project.addPortAtLocation(330,200);

   See also SonnetProject/addPort, SonnetProject/addPortToPolygon,
            SonnetProject/addPortCocalibrated, SonnetProject/addPortStandard,
            SonnetProject/addPortAutoGrounded



addPortAutoGrounded   Add a port to the project
 Port=Project.addPortAutoGrounded(...) will add an autogrounded port
 to the array of ports. A reference to the new port is returned.

   It requires the following arguments:
     1)  The Polygon to which the port is attached (or its debugID)
     2)  The Vertex to which the polygon is attached. The vertex number
           should be the index for the first vertex number that
           defines the polygon edge; if the user would like to
           attach a port between the third and fourth (X,Y)
           coordinate points for a polygon then the vertex
           number should be three. The port number for the
           port will be 'PortNumber'.
     3)  The Resistance for the port
     4)  The Reactance for the port
     5)  The Inductance for the port
     6)  The capacitance for the port
     7)  A character string which identifies a
           reference plane for the autogrounded port.
           this value is FIX for a reference
           plane and NONE for a calibration length.
     8)  A floating point number which provides the
           length of the reference plane when the type
           is FIX and provides the calibration length
           when the type is NONE.
     9) The Port Number(Optional)

   Note: This method is only for geometry projects.

   Example usage:
       % Add an autogrounded port
       Port=Project.addPortAutoGrounded(11,1,50,0,0,0,'FIX',10);

   See also SonnetProject/addPort, SonnetProject/addPortToPolygon,
            SonnetProject/addPortCocalibrated, SonnetProject/addPortAtLocation,
            SonnetProject/addPortStandard
```

```
addPortCocalibrated   Add a port to the project
 Port=Project.addPortCocalibrated(...) will add a standard port
 to the array of ports. A reference to the new port is returned.

   It requires the following arguments:
     1) The Polygon to which the port is attached (or its debugID)
     2) The Name of the group to which it belongs
     3)  The Vertex to which the polygon is attached. The vertex number
           should be the index for the first vertex number that
           defines the polygon edge; if the user would like to
           attach a port between the third and fourth (X,Y)
           coordinate points for a polygon then the vertex
           number should be three. The port number for the
           port will be 'PortNumber'.
     4) The Resistance for the port
     5) The Reactance for the port
     6) The Inductance for the port
     7) The capacitance for the port
     8) The Port Number (Optional)

   Note: This method is only for geometry projects.

   Example usage:
       % Add an co-calibrated port
       Port=Project.addPortCocalibrated(11,'A',1,75,0,0,0);

   See also SonnetProject/addPort, SonnetProject/addPortToPolygon,
            SonnetProject/addPortAutoGrounded, SonnetProject/addPortAtLocation,
            SonnetProject/addPortStandard



addPortOnlyComponent   Add a ports only component
   Project.addPortOnlyComponent(...) adds a ports only
   component to a geometry project.

   addPortOnlyComponent takes the following arguments:
     1) The component name (Ex: 'COMP1')
     2) Level number
     3) A nx2 matrix of the component port locations.
           The first row should be the first port's X value, then its Y value
           The second row should be the second port's X value, then its Y value
              etc.
     4) (Optional) The terminal width
           This value should be either
               - "Feed" to use the feedline width (Default)
               - "Cell" for one cell width
               - A number which represents a custom width

   Note: This method is only for geometry projects.
   Note: This method will add components to a project.
         To modify the value of a component use the
         modifyComponentValue method.

   Example usage:
       Project.addPortOnlyComponent('COM1',0,[104.5 156; 104.5 189])
       Project.addPortOnlyComponent('COM2',0,[104.5 156; 104.5 189],5)
       Project.addPortOnlyComponent('COM3',0,[104.5 156; 104.5 189],'Feed')
       Project.addPortOnlyComponent('COM4',0,[104.5 156; 104.5 189],'1Cell')



addPortStandard   Add a port to the project
   Port=Project.addPortStandard(Polygon,Vertex,Resistance,Reactance,
   Inductance,Capacitance) will add a standard port to the
```

```
   array of ports. The vertex number should be the index for
   the first vertex number that defines the polygon edge;
   if the user would like to attach a port between the
   third and fourth (X,Y) coordinate points for a polygon
   then the vertex number should be three. A reference to the
   new port is returned.

   Port=Project.addPortStandard(Polygon,Vertex,Resistance,Reactance,
   Inductance,Capacitance,PortNumber) will add a standard port to the
   array of ports. The vertex number should be the index for
   the first vertex number that defines the polygon edge;
   if the user would like to attach a port between the
   third and fourth (X,Y) coordinate points for a polygon
   then the vertex number should be three. The port number
   for the port will be 'PortNumber'. A reference to the new
   port is returned.

   Note: This method is only for geometry projects.

   Example usage:
       % Add a standard port
       Port=Project.addPortStandard(11,1,75,0,0,0);

   See also SonnetProject/addPort, SonnetProject/addPortToPolygon,
            SonnetProject/addPortCocalibrated, SonnetProject/addPortAtLocation,
            SonnetProject/addPortAutoGrounded



addPortToPolygon   Add a port to the project
   Port=Project.addPortToPolygon(Polygon, Vertex) will add a
   standard port to the specified vertex of the passed
   polygon. The vertex number should be the index for
   the first vertex number that defines the polygon edge;
   if the user would like to attach a port between the
   third and fourth (X,Y) coordinate points for a polygon
   then the vertex number should be three. A reference to
   the new port is returned.

   Note: This method is only for geometry projects.

   Example usage:

       % In this example we will add a port to
       % a particular polygon in the project.
       % The X and Y coordinates of the sixth
       % polygon in the project are as follows:
       %
       % Project.getPolygon(6).XCoordinateValues
       % ans =
       %    [34]    [227]    [227]    [34]    [34]
       %
       % Project.getPolygon(6).YCoordinateValues
       % ans =
       %    [105]    [105]    [75]    [75]    [105]
       %
       % Add we want to add a port on the edge between (227,105)
       % and (227,75). Because (227,105) is the second coordinate
       % pair the vertex number should be two. The polygon
       % in this case is the sixth polygon in the project; we can
       % get a reference to the sixth polygon in the project
       % with the command Project.getPolygon(6).
       PortReference=Project.addPort(6,2);

   See also SonnetProject/addPort, SonnetProject/addPortAtLocation,
            SonnetProject/addPortCocalibrated, SonnetProject/addPortStandard,
            SonnetProject/addPortAutoGrounded
```

```
addProjectFileElement   Creates a project file element
   Project.addProjectFileElement(File,PortNodes,SweepFromSubproject)
   Will add an project file to the circuit connected to the ports
   specified by PortNodes. SweepFromSubproject should be either 0 or 1.
   0 to indicate that you use the sweep from this project or 1 to
   indicate that you use the sweep from the subproject.

   Project.addProjectFileElement(File,PortNodes,SweepFromSubproject,Network)
   Will add an project file to the circuit connected to the ports
   specified by PortNodes. SweepFromSubproject should be either 0 or 1.
   0 to indicate that you use the sweep from this project or 1 to
   indicate that you use the sweep from the subproject. The network
   selection may be the network's index or the network's name.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a project file element to the first network of the project
       Project.addProjectFileElement('projectFile.son',[1,2],0);

       % Add a project file element to the second network of the project
       Project.addProjectFileElement('projectFile.son',[1,2],0,2);

   See also SonnetProject/addResistorElement,
            SonnetProject/addInductorElement,
            SonnetProject/addCapacitorElement,
            SonnetProject/addTransmissionLineElement,
            SonnetProject/addPhysicalTransmissionLineElement,
            SonnetProject/addDataResponseFileElement,
            SonnetProject/addNetworkElement



addReferencePlane   Adds a reference plane to the project
   Project.addReferencePlane(...) will add another reference plane
   to the array of reference planes.

   addReferencePlane requires these arguments:
     1) The Side    -  the side the plane is on ('LEFT', 'RIGHT', 'Top', 'BOTTOM')
     2) The Type    -  type of reference plane (FIX, LINK, NONE)
     3) The length  -  length of the reference plane (If type is FIX or NONE)
          or
     3) The polygon -  the polygon to which the reference plane is linked
                       either the polygon object or the polygon's ID.
     4) If it is a polygon the vertex to which the reference
         plane will be connected to will need to be specified

   Note: This method is only for geometry projects.

   Example usage:

       % Add a reference plane to the 'TOP' side
       % of type 'FIX' of length 12.
       Project.addReferencePlane('TOP','FIX',12);

       % Add a reference plane to the 'BOTTOM' side
       % of type 'NONE' of length 10.
       Project.addReferencePlane('BOTTOM','NONE',10);

       % Add a reference plane to the 'RIGHT' side
       % of type 'LINK' with vertex 1 of a particular polygon.
       Project.addReferencePlane('RIGHT','LINK',aPolygonObject,1);
```

```
        % Add a reference plane to the 'RIGHT' side
        % of type 'LINK' at the 2nd vertex of the polygon
        % with an ID of 1.
        Project.addReferencePlane('RIGHT','LINK',1,2);
```

```
addReferencePlaneToPortGroup   Adds a reference plane to a cocalibrated port group
   Project.addReferencePlaneToPortGroup(...) will add a reference plane
   to a cocalibrated port group.

   addReferencePlaneToPortGroup requires these arguments:
     1) The Name    -  the name of the cocalibrated port group
     2) The Side    -  the side the plane is on ('LEFT', 'RIGHT', 'Top', 'BOTTOM')
     3) The Type    -  type of reference plane (FIX, LINK, NONE)
     4) The length  -  length of the reference plane (If type is FIX or NONE)
          or
     4) The polygon -  the polygon to which the reference plane is linked
                       either the polygon object or the polygon's ID.
     5) If it is a polygon the vertex to which the reference
         plane will be connected to will need to be specified

   Note: This method is only for geometry projects.

   Example usage:

       % Add a reference plane to the 'TOP' side
       % of type 'FIX' of length 12.
       Project.addReferencePlaneToPortGroup('A','TOP','FIX',12);

       % Add a reference plane to the 'BOTTOM' side
       % of type 'NONE' of length 10.
       Project.addReferencePlaneToPortGroup('A','BOTTOM','NONE',10);

       % Add a reference plane to the 'RIGHT' side
       % of type 'LINK' with vertex 1 of a particular polygon.
       Project.addReferencePlaneToPortGroup('B','RIGHT','LINK',aPolygonObject,1);

       % Add a reference plane to the 'RIGHT' side
       % of type 'LINK' at the 2nd vertex of the polygon
       % with an ID of 1.
       Project.addReferencePlaneToPortGroup('B','RIGHT','LINK',1,2);
```

```
addResistorComponent   Add a resistor component
   aComponent=Project.addResistorComponent(...) adds an ideal resistor
   component to a geometry project.  A reference to the newly added
   component is returned which can be used to modify the component's
   settings.

   addResistorComponent takes the following arguments:
     1) The component name (Ex: 'R1')
     2) The resistor value (Ex: 50)
     3) Level number
     4) A nx2 matrix of the component port locations.
           The first row should be the first port's X value, then its Y value
           The second row should be the second port's X value, then its Y value
              etc.
     5) (Optional) The terminal width
           This value should be either
               - "Feed" to use the feedline width (Default)
               - "Cell" for one cell width
               - A number which represents a custom width

   Note: This method is only for geometry projects.
   Note: This method will add components to a project.
```

```
          To modify the value of a component use the
          modifyComponentValue method.

   Example usage:
       Project.addResistorComponent('R1',50,0,[104.5 156; 104.5 189])
       Project.addResistorComponent('R2',50,0,[104.5 156; 104.5 189],5)
       Project.addResistorComponent('R3',50,0,[104.5 156; 104.5 189],'Feed')
       Project.addResistorComponent('R4',50,0,[104.5 156; 104.5 189],'1Cell')



addResistorElement   Creates a resistor element
   Project.addResistorElement(Node1,Node2,Resistance) will add
   an resistor element to the circuit between Node1 and Node2 with
   the specified resistance. If the second node of the resistor
   should not be attached to any node then Node2 should be [].

   Project.addResistorElement(Node1,Node2,Resistance,Network) will add
   an resistor element to the specified network of the circuit between
   Node1 and Node2 with the specified resistance. If the second node
   of the resistor should not be attached to any node then Node2 should
   be []. The network selection may be the network's index or the
   network's name.

   Note: This method is only for netlist projects.

   Example usage:

       % Add a resistor element to the first network
       % in the project. The resistor is connected
       % from node 1 to 2 with resistance of 50
       Project.addResistorElement(1,2,50);

       % Add a resistor element to the second network
       % in the project. The resistor is connected
       % from node 1 to 2 with resistance of 50
       Project.addResistorElement(1,2,50,2);

   See also SonnetProject/addInductorElement,
            SonnetProject/addCapacitorElement,
            SonnetProject/addTransmissionLineElement,
            SonnetProject/addPhysicalTransmissionLineElement,
            SonnetProject/addDataResponseFileElement,
            SonnetProject/addProjectFileElement,
            SonnetProject/addNetworkElement



addSimpleFrequencySweep   Adds an 'SIMPLE' type of sweep to the project
   Project.addSimpleFrequencySweep(StartFrequency,EndFrequency,StepValue) adds
   a 'SIMPLE' type of frequency sweep to the project.

   This function will change the selected frequency sweep to 'SIMPLE'.

   Example usage:

       % Add an 'SIMPLE' type of sweep to the project.
       % the sweep will go from 5 to 10 with steps of 1.
       Project.addSimpleFrequencySweep(5,10,1);

 See also SonnetProject/addFrequencySweep



addStepFrequencySweep   Adds an 'STEP' type of sweep to the project
   Project.addStepFrequencySweep(Frequency) adds a 'STEP' type of
   frequency sweep to the project.
```

```
   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add an 'STEP' type of sweep to the project.
       % the sweep simulate at frequency 5
       Project.addStepFrequencySweep(5);

 See also SonnetProject/addFrequencySweep



addSweepFrequencySweep   Adds a 'Sweep' type of sweep to the project
   Project.addSweepFrequencySweep(StartFrequency,EndFrequency,StepFrequency)
   adds a 'SWEEP' type of frequency sweep to the project.

   This sweep is part of a combination frequency sweep.
   This function will change the selected frequency sweep
   to frequency sweep combination.

   Example usage:

       % Add a 'Sweep' type of sweep to the project.
       % the sweep will go from 5 to 10 in steps of 1.
       Project.addSweepFrequencySweep(5,10,1);

 See also SonnetProject/addFrequencySweep



addSymmetricDimensionParameter   Adds a dimension parameter
   Project.addSymmetricDimensionParameter(...) will add a symmetric
   geometry dimension parameter to the project.

   addSymmetricDimensionParameter ten arguments:
     1) The parameter name (Ex: 'Width')
     2) Handle for first reference polygon or the polygon's ID
     3) The vertex number used for the first reference polygon
     4) Handle for second reference polygon or the polygon's ID
     5) The vertex number used for the second reference polygon
     6) A cell array of any polygons that have points that should
           be included in the first point set. If there is
           only one polygon to be altered then this parameter
           does not need to be a cell array. Polygons in the
           first point set are the ones to be altered in the
           same way as the first reference point.
     7) A cell array of vectors that indicate which polygon
           vertices should be in the first point set. If there is
           only one polygon to be altered then this parameter
           does not need to be a cell array.
     8) A cell array of any polygons that have points that should
           be included in the second point set. If there is
           only one polygon to be altered then this parameter
           does not need to be a cell array. Polygons in the
           second point set are the ones to be altered in the
           same way as the first reference point.
     9) A cell array of vectors that indicate which polygon
           vertices should be in the first point set. If there is
           only one polygon to be altered then this parameter
           does not need to be a cell array.
    10) The direction of movement; this may be 'x', 'X', or 'XDir'
           for the X direction and 'y', 'Y', or 'YDir' for the Y
           direction.
    11) (Optional) The equation that should be used.
```

```
Note: This method is only for geometry projects.
Note: This method will add dimension parameters to a project.
      To modify the value of a dimension parameter use the
      modifyVariableValue method.

Example usage:

  Example 1:
    % We have a polygon in a project and we want to alter its
    % width with a dimension parameter. This particular polygon
    % has coordinate values of: (10,10),(30,10),(30,40),
    % (10,40),(10,10). The polygon has an ID of seven. The polygon
    % looks like the following diagram with the vertices numbered.
    % We want the polygon to grow/shrink on both the left and right
    % hand sides.
    %
    %                          4-----3
    %                          |     |
    %                          |     |
    %                          |     |
    %                          1-----2
    %
    % To accomplish our goal we can add a symmetric dimension parameter
    % to the project. The parameter will be named 'Width' and
    % be attached to the polygon with an ID of seven. The first
    % reference vertex will be the first vertex of the desired
    % polygon and the second vertex value will be the second
    % vertex of the polygon. The two reference points signify
    % a move in the X direction.
    %
    % Now we will add some polygons that have altering points
    % to the point set.  In this case we want the left two coordinates
    % to move together (coordinates 1 and 4) and the right two
    % coordinates to move together (coordinates 2 and 3). Each set
    % of points that will move together is a point set. One of the
    % point sets should be [1 4] and the other [2 3]. Alternatively
    % the point sets may just be [4] and [3] because points 1 and 2 are
    % already going to be moved because they are the reference points.

    Project.addSymmetricDimensionParameter('Width',7,1,7,2,7,[1 4],7,[2 3],'x');

    % Alternately, the polygon's coordinates could have been selected
    % easier with the polygon methods lowerRightVertex(), lowerLeftVertex(),
    % upperRightVertex(), upperLeftVertex(). These methods will return
    % the index of the coordinate that is at the desired location of
    % the polygon. The polygon coordinate methods are intended for
    % rectangular polygons only. Using the polygon coordinate access
    % methods on non-rectangular polygons could potential yield
    % undesirable results (Example: what is the lower left corner
    % of a spiral? lowerLeftVertex() will return the best value it
    % can but the user should be aware that in that case they may
    % be better off specifying the coordinate manually). In order
    % to use methods such as lowerRightVertex() we will need to
    % obtain a reference to the desired polygon; this can be
    % accomplished using the findPolygonUsingId() method.

    [~, polygon]=Project.findPolygonUsingId(7);
    Project.addSymmetricDimensionParameter('Width',...
            polygon,polygon.lowerLeftVertex(),...
            polygon,polygon.lowerRightVertex(),...
            polygon,polygon.upperLeftVertex(),...
            polygon,polygon.upperRightVertex(),'x');

  Example 2:
    % We have two polygons in a project and we want to alter
    % their separation with a dimension parameter. The left
```

```
  % polygon has coordinate values of: (10,10),(30,10),(30,40),
  % (10,40),(10,10). The right polygon has coordinate values
  % of (50,10),(80,10),(80,30),(70,30),(70,40),(50,40). The left polygon
  % has an ID of seven and the right polygon has an ID of eight.
  % The polygon layout looks like the following diagram with the vertices
  % numbered. We want to alter the separation between the polygons such
  % that they are closer together / farther apart.
  %
  %             4-----3       6-----5
  %             |     |       |     |__3
  %             |     |       |     4  |
  %             |     |       |        |
  %             1-----2       1--------2
  %
  % To accomplish our goal we will add a dimension parameter. We
  % will call our parameter 'Sep'. The first reference point will
  % be attached to vertex number two of the left polygon (ID of seven)
  % and the second reference point will be attached to vertex
  % number one of the right polygon (ID of eight).
  %
  % In this example we want to alter all the points for the left
  % polygon separately and all the points in the right polygon
  % separately. This can be done by making then be in different
  % point sets. We may indicate that all the points in a polygon
  % should be altered by passing [] for the vertex vector.

  Project.addSymmetricDimensionParameter('Sep',7,2,8,1,7,[],8,[],'x');

Example 3:
  % We have three polygons in a project and we want to alter the separation
  % between the right two polygons and the left most polygon. The left
  % polygon has coordinate values of: (10,10),(30,10),(30,40),
  % (10,40),(10,10). The middle polygon has coordinate values
  % of (50,10),(80,10),(80,30),(70,30),(70,40),(50,40). The polygon on
  % the right has coordinate values of (90,10),(120,10),(120,30),(110,30),
  % (110,40),(90,40). The left polygon has an ID of seven, the middle
  % polygon has an ID of eight  and the right polygon has an ID of nine.
  % The polygon layout looks like the following diagram with the vertices
  % numbered. We want the middle and right polygons to move closer or farther
  % away from the fixed left polygon.
  %
  %             4-----3         6-----5     6-----5
  %             |     |         |     |__3  |     |__3
  %             |     |         |     4  |  |     4  |
  %             |     |         |        |  |        |
  %             1-----2         1--------2  1--------2
  %                   |<--Sep-->|
  %
  % To accomplish our goal we will add a dimension parameter. We
  % will call our parameter 'Sep'. The first reference point will
  % be attached to vertex number two of the left polygon (ID of seven)
  % and the second reference point will be attached to vertex
  % number one of the middle polygon (ID of eight).
  %
  % Now we will add some polygons that have altering points
  % to the point sets. In this case we want the left polygon to
  % move independently and the right two polygons to move
  % together. So the left most polygon should be used for the
  % first point set and the right two polygons used for the
  % second point set. Because the second point set contains more
  % than one polygon the polygons and vertices must be specified
  % as cell arrays. Because the entire polygons should be moved the
  % vertices may be specified by the empty set ([]); in this example
  % we will explicitly state the vertices anyway so that the user can
  % see how to indicate individual vertices.

  aPointSet1Polygons=7;
```

```
        aPointSet1Points=[1 2 3 4];
        aPointSet2Polygons{1}=8;
        aPointSet2Polygons{2}=9;
        aPointSet2Points{1}=[1 2 3 4 5 6];
        aPointSet2Points{2}=[1 2 3 4 5 6];

Project.addAnchoredDimensionParameter('Sep',7,2,8,1,aPointSet1Polygons,aPointSet1Points,aPointSet
2Polygons,aPointSet2Points,'x');
```

```
 addTouchstoneOutput   Find a port given an approximate location
    Project.addTouchstoneOutput() will add a touchstone
    file output to the project. The output file will have
    the same base filename as the project but will have the
    extension ".s#p" where # is the number of ports currently
    in the project.

    Note: This method is the equivalent of the following command
          Project.addFileOutput('TS','D','Y','$BASENAME.s#p','IC','N','S','MA','R',50);
          where # is the number of ports in the project.

  See also SonnetProject/findPortUsingPoint
```

```
 addTransmissionLineElement   Creates a transmission line element
    Project.addTransmissionLineElement(...) will add an transmission line to the circuit

    Project.addTransmissionLineElement(Node1,Node2,Impedance,Length,Frequency)
    will add a transmission line element to the circuit between Node1
    and Node2 with the specified impedance, length and frequency of operation.
    If the second node of the capacitor should not be attached to any node
    then Node2 should be [].

    Project.addTransmissionLineElement(Node1,Node2,Impedance,Length,Frequency,Network)
    will add a transmission line element to the circuit between Node1
    and Node2 with the specified impedance, length and frequency of operation.
    If the second node of the capacitor should not be attached to any node
    then Node2 should be []. The network selection may be the network's index
    or the network's name.

    Note: This method is only for netlist projects.

    Example usage:

        % Add a transmission line element to
        % the first network of the project
        % connected from node 1 to 2 with
        % an impedance of 100, an electrical
        % length of 1000 and a frequency of 10.
        Project.addTransmissionLineElement(1,2,100,1000,10);

        % Add a transmission line element to
        % the second network of the project
        % connected from node 1 to 2 with
        % an impedance of 100, an electrical
        % length of 1000 and a frequency of 10.
        Project.addTransmissionLineElement(1,2,100,1000,10,2);

    See also SonnetProject/addResistorElement,
             SonnetProject/addInductorElement,
             SonnetProject/addCapacitorElement,
             SonnetProject/addPhysicalTransmissionLineElement,
             SonnetProject/addDataResponseFileElement,
             SonnetProject/addProjectFileElement,
             SonnetProject/addNetworkElement
```

```
addVariableSweepSimple   Add a variable sweep
   Project.addVariableSweep(theFreqSweepHandle) will add a variable sweep
   to the array of sweep entries. The specified frequency sweep will
   be used for the parameter sweep

   The supplied sweep type must be one of the following:
       ABS_ENTRY   -   Adaptive Band Synthesis Sweep
       ABS_FMAX    -   Find the maximum frequency response.
       ABS_FMIN    -   Find the minimum frequency response.
       DC_FREQ     -   Analyze at a DC frequency point.
       STEP        -   Discrete analysis frequency
       SWEEP       -   Linear frequency sweep with stated interval.
       ESWEEP      -   Exponential frequency sweep.
       LSWEEP      -   Linear frequency sweep with number of points.

   Example usage:
       % Create an ABS frequency sweep object
       aSweep=SonnetFrequencyAbsEntry();
       aSweep.StartFreqValue=4.5;
       aSweep.EndFreqValue=5.5;

       % Create a variable sweep from
       % the ABS frequency sweep.
       Project.addVariableSweep(aSweep);



addVariableSweepParameter   Add a variable sweep
   Project.addVariableSweepParameter(...) will add a variable
   sweep parameter to the array of sweep entries.

   Input arguments are:
       1) Parameter Name -- The name of the parameter to sweep
       2) Min Value -- Starting value of the sweep
       3) Max Value -- Ending value of the sweep
       4) Number of Points -- Number of points on the sweep.
             for a corner sweep make this value be an empty matrix.
       5) Sweep Index (Optional) -- The index for the variable
              sweep entry block this parameter should be added
              to. Default is the first.

   Note: The specified variable name should already be defined
         and incorporated into the project or Sonnet will
         not be able to perform the simulation.

   Example usage:
       % Add an ABS sweep of variable 'VAR' with a minimum
       % of 5 max of 10 simulating 15 points.
       Project.addVariableSweepSimple('VAR',5,10,15)



addViaPolygon   Add a via polygon to the polygon array
   Project.addViaPolygon(...) will add a Via Polygon
   to the array of Polygons.

   addViaPolygon requires these arguments:
     1)  The level the VIA attaches to.
     2)  metallization  Level Index (The level the polygon is on)
     3)  The type of metal used for the polygon. This may either
          be a the index for the metal type in the array of
          metal types, or the name of the metal type
          (Ex: 'Copper'). Lossless metal is not in the array
          of metals but can be selected by either passing 0
```

```
          or 'Lossless'.
      4)  A string to identify the fill type used for the polygon.
            N indicates staircase fill, T indicates diagonal
            fill and V indicates conformal mesh. Note that filltype
            only applies to metal
            polygons; this field is ignored for dielectric brick polygons
      5)  Minimum subsection size in X direction
      6)  Minimum subsection size in Y direction
      7)  Maximum subsection size in X direction
      8)  Maximum subsection size in Y direction
      9)  The Maximum Length for The Conformal Mesh Subsection
      10) Edge mesh setting. Y indicates edge meshing is on for this
            polygon. N indicates edge meshing is off.
      11) A matrix for the X coordinate values.
      12) A matrix for the Y coordinate values

   Note: Many users will prefer to use the 'addViaPolygonEasy' method.
   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:
       % Create a via at level 0, attached to 'GND', metal type -1 (lossless),
       % staircase fill, X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
       Project.addViaPolygon('GND',0,0,'N',0,0,50,100,0,'Y',x,y);

       % Create a via at level 0, attached to 'GND', metal type 'Copper',
       % staircase fill, X subsection size from 0 to 50,
       % Y subsection size from 0 to 100.
       x=[5,10,10,5,5];
       y=[10,10,20,20,10];
       Project.addViaPolygon('GND',0,'Copper','N',0,0,50,100,0,'Y',x,y);

 See also SonnetProject/addViaPolygonEasy



addViaPolygonEasy   Add a via polygon to the polygon array
   Polygon=Project.addViaPolygonEasy(...) will add an via polygon
   to the array of polygons. A reference to the polygon
   is returned.

   addViaPolygonEasy requires these arguments:
      1) metallization  Level Index (The level the polygon is on)
      2) The level the via is connected to
      3) A matrix for the X coordinate values
      4) A matrix for the Y coordinate values
      5) (Optional) The type of metal used for the polygon.
           This may either be a the index for the metal
           type in the array of metal types, or the name
           of the metal type (Ex: 'Copper'). If this value
           is not specified then lossless metal will be used.

   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:
       % Lossless via at level 0, attached to 'GND'
       Project.addViaPolygonEasy(0,'GND',[5,10,10,5,5],[10,10,20,20,10]);

       % Copper via at level 0, attached to 'GND' (Copper
```

```
       % metal must type must be defined for the project)
       Project.addViaPolygonEasy(0,'GND',[5,10,10,5,5],[10,10,20,20,10],'Copper');

 See also SonnetProject/addViaPolygon



assignAllPolygonssequentialIds  Makes sure polygons have unique IDs
   Project.assignAllPolygonssequentialIds() will make sure all the
   polygons in a project have unique debugIds by making their
   debugIds be their index in the array of polygons. The
   debugIds of all the polygons in the project may be changed.

   Note: This method is only for geometry projects.

 See also SonnetProject/assignUniqueDebugId,
          SonnetProject/generateUniqueId



assignUniqueDebugId   Assign a polygon an unique debugId.
   Project.assignUniqueDebugId(aPolygon) will assign the
   passed polygon a unique debugId. The passed polygon
   does not necessarily need to be from same project.

   Note: This method is only for geometry projects.

 See also SonnetProject/assignAllPolygonsSequentialIds,
          SonnetProject/generateUniqueId



changeAngleUnit   Change project's angle unit
   Project.changeAngleUnit(string) modifies the angle unit
   selected for the project. The passed angle unit should
   be a unit that is supported by Sonnet.
   (At the moment the only supported unit is DEG)

   changeAngleUnit(unitString)     Changes the selected angle unit
                                   to the passed unit identifier

   Example usage:

       % Change the angle unit to 'DEG'
       Project.changeAngleUnit('DEG');



changeBottomCover   Changes the type for the bottom cover
   changeBottomCover(Project,theType) will modify the
   cover to be the specified cover type. The cover
   type may be one of the three built in cover types
   that are defined for all Sonnet projects
   ('Lossless','Freespace','WG Load') or the name
   of an existing user defined metal type (the metal
   type must already be defined).

   Note: This method is only for geometry projects.

   Example usage:

       % Make a new Sonnet project and
       % make the cover be 'Freespace'
       theProject=SonnetProject();
       Project.changeBottomCover('Freespace');

       % Modify the cover to be a custom
```

```
        % user defined type known as 'ThinCopper'
        Project.changeBottomCover('ThinCopper');

    See also SonnetProject/changeTopCover



changeBoxSize   Changes the size of the box
    Project.changeBoxSize(XSize,YSize) changes the
    size of the Sonnet box. The Sonnet box encompasses
    the circuit area.  The new box width will be
    XSize and the new box height will be YSize.

    Note: This function is the same as changeBoxSizeXY
    Note: This method is only for geometry projects.

    See also SonnetProject/changeBoxSizeXY         SonnetProject/changeBoxSizeX
             SonnetProject/changeBoxSizeY          SonnetProject/changeNumberOfCells
             SonnetProject/changeNumberOfCellsXY   SonnetProject/changeNumberOfCellsX
             SonnetProject/changeNumberOfCellsY



changeBoxSizeX   Changes the size of the box
    Project.changeBoxSizeX(XSize) changes the size of the Sonnet box
    in the X direction only. The Sonnet box encompasses
    the circuit area. The new box width will be XSize.

    Note: This method is only for geometry projects.

    See also SonnetProject/changeBoxSize,         SonnetProject/changeBoxSizeXY,
             SonnetProject/changeBoxSizeY,        SonnetProject/changeNumberOfCells,
             SonnetProject/changeNumberOfCellsXY, SonnetProject/changeNumberOfCellsX,
             SonnetProject/changeNumberOfCellsY



changeBoxSizeXY   Changes the size of the box
    Project.changeBoxSizeXY(XSize,YSize) changes the size of the Sonnet box.
    The Sonnet box encompasses the circuit area. The new box width will be
    XSize and the new box height will be YSize.

    Note: This method is only for geometry projects.
    Note: This function is the same as changeBoxSize

    See also SonnetProject/changeBoxSize,         SonnetProject/changeBoxSizeX,
             SonnetProject/changeBoxSizeY,        SonnetProject/changeNumberOfCells,
             SonnetProject/changeNumberOfCellsXY, SonnetProject/changeNumberOfCellsX,
             SonnetProject/changeNumberOfCellsY



changeBoxSizeY   Changes the size of the box
    Project.changeBoxSizeY(YSize) changes the size of the Sonnet box
    in the Y direction only. The Sonnet box encompasses
    the circuit area. The new box height will be YSize.

    Note: This method is only for geometry projects.

    See also SonnetProject/changeBoxSize,         SonnetProject/changeBoxSizeXY,
             SonnetProject/changeBoxSizeX,        SonnetProject/changeNumberOfCells,
             SonnetProject/changeNumberOfCellsXY, SonnetProject/changeNumberOfCellsX,
             SonnetProject/changeNumberOfCellsY



changeCapacitanceUnit   Change project's capacitance unit
```

```
   Project.changeCapacitanceUnit(string) modifies the capacitance unit
   selected for the project. The passed capacitance unit should
   be a unit that is supported by Sonnet.

   changeCapacitanceUnit(unitString)     Changes the selected capacitance
                                         unit to the passed unit identifier
   Example usage:

       % Change the resistance unit to 'nF'
       Project.changeCapacitanceUnit('nF');



changeCellSizeUsingBoxSize   Changes the cell size
   Project.changeCellSizeUsingBoxSize(XCellSize,YCellSize) changes
   the cell size used for a project. The size of the box in each
   direction will be modified to realize the given
   cell size.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeCellSizeUsingBoxSizeXY

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY



changeCellSizeUsingBoxSizeX   Changes the cell size
   Project.changeCellSizeUsingBoxSizeX(XCellSize) changes the cell size
   used for a project. The box size in the X direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY



changeCellSizeUsingBoxSizeXY   Changes the cell size
   Project.changeCellSizeUsingBoxSizeXY(XCellSize,YCellSize) changes the cell size
   used for a project. The size of the box in each
   direction will be modified to realize the given
   cell size.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeCellSizeUsingBoxSize

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY



changeCellSizeUsingBoxSizeY   Changes the cell size
   Project.changeCellSizeUsingBoxSizeY(YCellSize) changes the cell size
   used for a project. The box size in the Y direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
```

```
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY


changeCellSizeUsingNumberOfCells   Changes the cell size
   Project.changeCellSizeUsingNumberOfCells(XCellSize,YCellSize) changes the
   cell size used for a project. The number of cells in each direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeCellSizeUsingNumberOfCellsXY.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY


changeCellSizeUsingNumberOfCellsX   Changes the cell size
   Project.changeCellSizeUsingNumberOfCellsX(XCellSize) changes
   the cell size used for a project. The number of cells in the X direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY


changeCellSizeUsingNumberOfCellsXY   Changes the cell size
   Project.changeCellSizeUsingNumberOfCellsXY(XCellSize,YCellSize) changes
   the cell size used for a project. The number of cells in each direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeCellSizeUsingNumberOfCells

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY


changeCellSizeUsingNumberOfCellsY   Changes the cell size
   Project.changeCellSizeUsingNumberOfCellsY(YCellSize) changes the cell size
   used for a project. The number of cells in the Y direction
   will be modified to realize the given cell size.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY


changeConductivityUnit   Change project's conductivity unit
   Project.changeConductivityUnit(string) modifies the conductivity unit
   selected for the project. The passed conductivity unit should
```

```
   be a unit that is supported by Sonnet.

   changeConductivityUnit(unitString)     Changes the selected conductivity
                                          unit to the passed unit identifier



changeDielectricLayerThickness   Changes layer thickness
   Project.changeDielectricLayerThickness(N,Thickness) will change
   the thickness of the Nth dielectric layer.

   Project.changeDielectricLayerThickness(Name,Thickness) will change
   the thickness of the dielectric layer with the specified name.
   If none of the layers in the project have the specified name
   then an error will be thrown.

   Note: This method is only for geometry projects.

   Example usage:

       % Change the thickness of the first layer
       % to be 50 units thick.
       Project.changeDielectricLayerThickness(1,50)

 See also SonnetProject/replaceDielectricLayer



changeFrequencyUnit   Change project's frequency unit
   Project.changeFrequencyUnit(string) modifies the frequency unit
   selected for the project. The passed frequency unit should
   be a unit that is supported by Sonnet. (HZ, KHZ, MHZ, GHZ, THZ, PHZ)

   changeFrequencyUnit(unitString)     Changes the selected frequency unit
                                       to the passed unit identifier

   Example usage:

       % Change the frequency unit to 'HZ'
       Project.changeFrequencyUnit('HZ');



changeInductanceUnit   Change project's inductance unit
   Project.changeInductanceUnit(string) modifies the inductance unit
   selected for the project. The passed inductance unit should
   be a unit that is supported by Sonnet. (H, MH, UH, NH, PH, FH)

   changeInductanceUnit(unitString)     Changes the selected inductance unit
                                        to the passed unit identifier

   Example usage:

       % Change the inductance unit to 'H'
       Project.changeInductanceUnit('H');



changeLengthUnit   Change project's length unit
   Project.changeLengthUnit(string) modifies the length unit
   selected for the project. The passed length unit should
   be a unit that is supported by Sonnet. (MIL, UM, MM, CM, IN, M)

   changeLengthUnit(unitString)     Changes the selected length unit
                                    to the passed unit identifier

   Example usage:
```

```
       % Change the length unit to 'MIL'
       Project.changeLengthUnit('MIL');
```

changeMeshToCoarseWithEdgeMesh Changes memory/speed option
   This function will change a project's subsectioning
   setting to be course meshing with edge meshing

changeMeshToCoarseWithNoEdgeMesh Changes memory/speed option
   This function will change a project's subsectioning
   setting to be course meshing with no edge meshing

changeMeshToFineWithEdgeMesh Changes memory/speed option
   This function will change a project's subsectioning
   setting to be fine meshing with edge meshing

changeNumberOfCells Changes the number of cells
   Project.changeNumberOfCells(XCells,YCells) changes the number of cells
   that make up the grid. This function changes the
   number of cells in the X direction to be XCells
   and the number of cells in the Y direction to be YCells.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeNumberOfCellsXY

   See also SonnetProject/changeBoxSize, SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX, SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCellsXY, SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY

changeNumberOfCellsX Changes the number of cells
   Project.changeNumberOfCellsX(XCells) changes the number of cells
   that make up the grid. This function modifies the
   number of cells in the X direction to be XCells.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize, SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX, SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells, SonnetProject/changeNumberOfCellsXY,
            SonnetProject/changeNumberOfCellsY

changeNumberOfCellsXY Changes the number of cells
   Project.changeNumberOfCellsXY(XCells,YCells) changes the number of cells
   that make up the grid. This function changes the
   number of cells in the X direction to be XCells
   and the number of cells in the Y direction to be YCells.

   Note: This method is only for geometry projects.
   Note: This function is the same as changeNumberOfCells

   See also SonnetProject/changeBoxSize, SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX, SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells, SonnetProject/changeNumberOfCellsX,
            SonnetProject/changeNumberOfCellsY

```
changeNumberOfCellsY   Changes the number of cells
   Project.changeNumberOfCellsY(YCells) changes the number of cells
   that make up the grid. This function modifies the
   number of cells in the Y direction to be YCells.

   Note: This method is only for geometry projects.

   See also SonnetProject/changeBoxSize,        SonnetProject/changeBoxSizeXY,
            SonnetProject/changeBoxSizeX,       SonnetProject/changeBoxSizeY,
            SonnetProject/changeNumberOfCells,  SonnetProject/changeNumberOfCellsXY,
            SonnetProject/changeNumberOfCellsX



changePolygonType   Change the composition of a polygon
   Project.changePolygonType(ID,Type) will try to change the
   composition of the polygon with the debugID of ID to the
   passed type. If the polygon is a metal or via polygon then
   Type must be the name of a metal type in the project. If
   the polygon is a dielectric brick then Type must be the
   name of one of the brick types in the project.

   Project.changePolygonType(Polygon,Type) will try to change the
   composition of the passed polygon to the passed type.
   If the polygon is a metal or via polygon then Type
   must be the name of a metal type in the project. If
   the polygon is a dielectric brick then Type must be the
   name of one of the brick types in the project.

   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:

       % Change the polygon with debug ID 12 to 'ThinCopper'
       % (A metal type called 'ThinCopper' must already be
       % defined in the project).
       Project.changePolygonType(12,'ThinCopper');

       % Change the polygon with debug ID 12 to 'Lossless'
       % (Lossless is the default type for metal polygons).
       Project.changePolygonType(12,'Lossless');

 See also SonnetProject/changePolygonTypeUsingId,
          SonnetProject/changePolygonTypeUsingIndex



changePolygonTypeUsingId   Change the composition of a polygon
   Project.changePolygonType(ID,Type) will try to change the
   composition of the polygon with the debugID of ID to the
   passed type. If the polygon is a metal or via polygon then
   Type must be the name of a metal type in the project. If
   the polygon is a dielectric brick then Type must be the
   name of one of the brick types in the project.

   Project.changePolygonTypeUsingId(Polygon,Type) will try
   to change the composition of the passed polygon to the
   passed type. If the polygon is a metal or via polygon
   then Type must be the name of a metal type in the
   project. If the polygon is a dielectric brick then
   Type must be the name of one of the brick types
```

```
   in the project.

   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:

       % Change the polygon with debug ID 12 to 'ThinCopper'
       % (A metal type called 'ThinCopper' must already be
       % defined in the project).
       Project.changePolygonTypeUsingId(12,'ThinCopper');

       % Change the polygon with debug ID 12 to 'Lossless'
       % (Lossless is the default type for metal polygons).
       Project.changePolygonTypeUsingId(12,'Lossless');

 See also SonnetProject/changePolygonType,
          SonnetProject/changePolygonTypeUsingId



changePolygonTypeUsingIndex   Change the composition of a polygon
   Project.changePolygonType(N,Type) will try to change the
   composition of the Nth polygon in the array of polygons to the
   passed type. If the polygon is a metal or via polygon then
   Type must be the name of a metal type in the project. If
   the polygon is a dielectric brick then Type must be the
   name of one of the brick types in the project.

   Project.changePolygonTypeUsingIndex(Polygon,Type) will try
   to change the composition of the passed polygon to the
   passed type. If the polygon is a metal or via polygon
   then Type must be the name of a metal type in the
   project. If the polygon is a dielectric brick then
   Type must be the name of one of the brick types
   in the project.

   Note: This method is only for geometry projects.
   Note: Sonnet version 12 projects have a shared metal type for planar
         and via polygons. Sonnet version 13 projects have separate
         metal types for planar polygons and via polygons.

   Example usage:

       % Change first polygon in the array of polygons to
       % 'ThinCopper' (A metal type called 'ThinCopper'
       % must already be defined in the project).
       Project.changePolygonTypeUsingIndex(1,'ThinCopper');

       % Change first polygon in the array of polygons to
       % 'Lossless' (Lossless is the default type for
       %  metal polygons).
       Project.changePolygonTypeUsingIndex(1,'Lossless');

 See also SonnetProject/changePolygonType,
          SonnetProject/changePolygonTypeUsingIndex



changeResistanceUnit   Change project's resistance unit
   Project.changeResistanceUnit(string) modifies the resistance unit
   selected for the project. The passed resistance unit should
   be a unit that is supported by Sonnet. (OH, KOH, MOH)

   changeResistanceUnit(unitString)     Changes the selected resistance
```

unit to the passed unit identifier

Example usage:

```
% Change the resistance unit to 'OH'
Project.changeResistanceUnit('OH');
```

changeSelectedFrequencySweep Change project's selected frequency sweep
Project.changeSelectedFrequencySweep(string) modifies the selected frequency sweep for the project. The selected frequency sweep is the one that is performed for simulations. The selected frequency sweep should be a sweep that is recognized by Sonnet (ABS, SIMPLE, STD).

Example usage:

```
% Change the selected frequency sweep to adaptive band
Project.changeSelectedFrequencySweep('ABS');

% Change the selected frequency sweep to frequency combination
Project.changeSelectedFrequencySweep('STD');

% Change the selected frequency sweep to parameter sweep
Project.changeSelectedFrequencySweep('VARSWP');

% Change the selected frequency sweep to optimization sweep
Project.changeSelectedFrequencySweep('OPTIMIZE');
```

changeTopCover Changes the type for the top cover
changeTopCover(Project,theType) will modify the cover to be the specified cover type. The cover type may be one of the three built in cover types that are defined for all Sonnet projects ('Lossless','Freespace','WG Load') or the name of an existing user defined metal type (the metal type must already be defined).

Note: This method is only for geometry projects.

Example usage:

```
% Make a new Sonnet project and
% make the cover be 'Freespace'
theProject=SonnetProject();
Project.changeTopCover('Freespace');

% Modify the cover to be a custom
% user defined type known as 'ThinCopper'
Project.changeTopCover('ThinCopper');
```

See also SonnetProject/changeBottomCover

changeVariableSweepParameterState Modify variable sweep parameter status
Project.changeVariableSweepParameterState(VariableName) will modify the parameter in use value for the specified parameter in the first variable sweep.

Project.changeVariableSweepParameterState(VariableName,N) will modify the parameter in use value for the specified parameter in the Nth variable sweep.

Appropriate status values are 'N','Y','YN','YS', and 'YE'.

```
   See also SonnetProject/activateVariableSweepParameter



cleanOutputFiles   Deletes output files for a project
   Project.cleanOutputFiles() deletes any output response files present
   in the directory for a Sonnet project. cleanOutputFiles knows
   which files to delete by checking the fileoutBlock for the
   project to see if any output files are defined. If there are
   output files to be deleted then cleanOutputFiles will look
   for those files in the simulation directory and delete them if present.

   See also SonnetProject/cleanProject



cleanProject   Cleans a project
   Project.cleanProject() deletes the simulation data for the project.

   See also SonnetProject/cleanOutputFiles



 clone   Initializes a replica project
   newProject=Project.clone() will return a deep copy of a
   Sonnet project. The copy will have all the same values for
   the class properties but will contain completely separate
   handles.

   The new project will have no filename associated
   with it but it may be saved with the saveAs()
   command.

   Example usage:

       % Create a new Sonnet project object
       Project1=SonnetProject('project.son');

       % Clone the project
       Project2=Project1.clone();

       % Any modifications made to Project1
       % or Project2 will not affect the
       % other project.

 See also  SonnetProject/quickClone



compare   Compares two Sonnet Projects for equivalency
   [isEqual aOutput]=aFirstProject.compare(aSecondProject) compares
   the data stored in aFirstProject to the data for
   aSecondProject. isEqual is true if the two projects
   are the same. aOutput stores the data that came out
   of the comparison engine.

   [isEqual aOutput]=aFirstProject.compare(filename) compares
   the data stored in aFirstProject to the data for
   the Sonnet project represented by filename. isEqual
   is true if the two projects are the same. aOutput
   stores the data that came out of the comparison engine.

   Example usage:
       aFirstProject=SonnetProject('myProject1.son');
       aSecondProject=SonnetProject('myProject2.son');
       isEqual=aFirstProject.compare(aSecondProject);
```

```
 See also SonnetProject/addFrequencySweep



copyDielectricLayer    Copies a dielectric layer
   Project.copyDielectricLayer(N) makes a copy of the
   Nth dielectric layer and places it on the bottom
   of the stackup.

   Note: This method is only for geometry projects.

 See also SonnetProject/replaceDielectricLayer



copyMetalPolygon Makes a copy of a metal polygon
   newPolygonIndex=Project.copyMetalPolygon(index) makes a
   carbon copy of a metal polygon specified by an index
   in the array of polygons. The new polygon's index will
   be returned.

   Note: This method is only for geometry projects.

 See also SonnetProject/copyPolygon,
          SonnetProject/copyPolygonUsingId,
          SonnetProject/copyPolygonUsingIndex



copyPolygon      Makes a copy of a polygon and adds it to the project
   Polygon=Project.copyPolygon(ID) Returns a copy of the polygon with the
   passed ID value. The new polygon will have a unique ID.

   Polygon=Project.copyPolygon(Polygon) Returns a copy of the passed
   polygon. The new polygon will have a unique ID.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygonUsingId,
           SonnetProject/copyPolygonUsingIndex,
           SonnetProject/duplicatePolygon,
           SonnetProject/duplicatePolygonUsingId,
           SonnetProject/duplicatePolygonUsingIndex,



copyPolygonUsingId      Makes a copy of a polygon and adds it to the project
   Polygon=Project.copyPolygonUsingId(ID) Makes a copy of the polygon with the
   passed ID value. The new polygon will have a unique ID.

   Polygon=Project.copyPolygonUsingId(Polygon) Makes a copy of the passed
   polygon. The new polygon will have a unique ID.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygon,
           SonnetProject/copyPolygonUsingIndex,
           SonnetProject/duplicatePolygon,
           SonnetProject/duplicatePolygonUsingId,
           SonnetProject/duplicatePolygonUsingIndex,



copyPolygonUsingIndex      Makes a copy of a polygon and adds it to the project
   Polygon=Project.copyPolygonUsingIndex(N) Returns a copy of the Nth polygon
   in the array of polygons. The new polygon will have a unique debug ID.
```

```
   Polygon=Project.copyPolygonUsingIndex(Polygon) Returns a copy of the passed
   polygon. The new polygon will have a unique ID. The new polygon will be
   returned.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygon,
           SonnetProject/copyPolygonUsingId
           SonnetProject/duplicatePolygon,
           SonnetProject/duplicatePolygonUsingId,
           SonnetProject/duplicatePolygonUsingIndex,



deactivateVariableSweepParameter   Deactivates a variable sweep parameter
   Project.deactivateVariableSweepParameter(VariableName) will set the
   parameter in use value for the specified parameter in the
   first variable sweep to false.

   Project.deactivateVariableSweepParameter(VariableName,N) will
   set the parameter in use value for the specified parameter in the
   Nth variable sweep to false.

   See also SonnetProject/activateVariableSweepParameter



defineNewArrayMetalType   Defines a new type of via metal
   Project.defineNewArrayMetalType(Name,R,X) will define
   an array metal type for the project.

   Note: This method is only for geometry projects.
   Note: This method is only for Sonnet version 13 projects.

   Example usage:

       % Define a new array metal type named 'ArrayMetal1'
       Project.defineNewArrayMetalType('ArrayMetal1',50,100);

   See also SonnetProject/defineNewViaMetalType



defineNewBrickType   New anisotropic dielectric brick type
   Project.defineNewBrickType(...) will add a
   dielectric brick type to the array of brick types.

   There are two ways to use defineNewBrickType. The user
   may define a brick type using a set of custom options or
   the user may define a type using a predefined property set
   from the Sonnet library.

   If defineNewBrickType is used to import a brick type from the
   Sonnet library then the following arguments must be specified
     1) The name of the material

   defineNewBrickType can be used to add an Isotropic Dielectric
   brick type to the project by specifying the following parameters.
     1)  The name of the dielectric
     2)  Relative dielectric constant
     3)  Loss tangent
     4)  Bulk conductivity

   defineNewBrickType can be used to add an anisotropic Dielectric
   brick type to the project by specifying the following parameters.
     1)  The name of the dielectric
```

```
      2)  Relative dielectric constant in the X direction
      3)  Loss tangent in the X direction
      4)  Bulk conductivity in the X direction
      5)  Relative dielectric constant in the Y direction
      6)  Loss tangent in the Y direction
      7)  Bulk conductivity in the Y direction
      8)  Relative dielectric constant in the Z direction
      9)  Loss tangent in the Z direction
      10) Bulk conductivity in the Z direction

   Note: This method is only for geometry projects.

   Example usage:
       % Define the Aluminum Nitride brick material
       % using the Sonnet material library
       Project.defineNewBrickType('Aluminum Nitride');

       % Make a new brick material named 'Brick1' with
       % a relative dielectric constant of 1, a loss
       % tangent of 2 and a bulk conductivity of 3.
       Project.defineNewBrickType('Brick1',1,2,3);

       % Make a new brick material named 'Brick1' with
       % the following settings:
       % X direction:
       %   relative dielectric constant of 1
       %   loss tangent of 2
       %   bulk conductivity of 3
       % Y direction:
       %   relative dielectric constant of 4
       %   loss tangent of 5
       %   bulk conductivity of 6
       % Z direction:
       %   relative dielectric constant of 7
       %   loss tangent of 8
       %   bulk conductivity of 9
       Project.defineNewBrickType('Brick1',1,2,3,4,5,6,7,8,9);

 See also SonnetProject.addIsotropicDielectricBrickType



defineNewGeneralMetalType   Defines a new type of metal
   Project.defineNewGeneralMetalType(Name,Resistance,SkinCoefficient,
   Reactance,KineticInductance) will add a general type of
   metal to the array of metals.

   Note: This method is only for geometry projects.

   Example usage:

       % Define a new general metal type named 'GeneralMetal1'
       % with a resistance of 100, a skin coefficient of 50,
       % a reactance of 50 and a kinetic inductance of 50.
       Project.defineNewGeneralMetalType('GeneralMetal1',100,50,50,50);

   See also SonnetProject/defineNewMetalType



defineNewMetalType   Create a new type of metal
   Project.defineNewMetalType(...) will add
   a metal type to the project.

   There are two ways to use defineNewMetalType. The user
   may define a metal type using a set of custom options or
   the user may define a type using a predefined property set
```

```
from the Sonnet library.

If defineNewMetalType is to import a metal type from the
Sonnet library then the following arguments must be specified
  1) The name of the metal
  2) The metal's thickness

If defineNewMetalType is to be used to define a custom
metal type then the user must first specify the type of
metal that is being defined and then specify the parameters
for the metal type.

defineNewMetalType requires a type as
the first argument which should
be one of the following:

    NOR   -   Normal Metal
    RES   -   Resistor Metal
    NAT   -   Native Metal
    SUP   -   General Metal
    SEN   -   Sense Metal
    TMM   -   Thick Metal
    RUF   -   Rough Metal

Then you will need to supply the necessary
arguments for each type as follows:

NOR-Normal Metal
  1) The Name of the metal
  2) The Conductivity of the metal
  3) The Current Ratio of the metal
  4) The Thickness of the metal

RES-Resistor Metal
  1) The Name of the metal
  2) The Resistance of the metal

NAT-Native Metal
  1) The Name of the metal
  2) The Resistance of the metal
  3) The Skin Coefficient of the metal

SUP-General Metal
  1) The Name of the metal
  2) The Resistance of the metal
  3) The Skin Coefficient of the metal
  4) The Reactance of the metal
  5) The Kinetic Inductance of the metal

SEN-Sense Metal
  1) The Name of the metal
  2) The Reactance of the metal

TMM-Thick Metal
  1) The Name of the metal
  2) The Conductivity of the metal
  3) The Current Ratio of the metal
  4) The Thickness of the metal
  5) The Number of Sheets of the metal

RUF-Rough Metal
  1) The Name of the metal
  2) Whether the metal should be modeled as being thick or thin
       This value may be either (case insensitive)
           - 'thick' or 'THK' for thick
           - 'thin' or 'THN' for thin
  3) The Thickness of the metal
```

```
      4) The Conductivity of the metal
      5) The Current Ratio of the metal
      6) The Roughness of the top
      7) The Roughness of the bottom

   Note: This method is only for geometry projects.
   Note: For Sonnet 13 projects planar metal types are
         different than via metal types. For information
         on how to define via metal types see the help
         information for defineNewViaMetalType.

   Example usage:

       % Import aluminum from the Sonnet metal library
       Project.defineNewMetalType('Aluminum',1.4);

       % Define a new normal metal type named 'NormalMetal1'
       % of conductivity 58000000, current ratio 50 and thickness 50.
       Project.defineNewMetalType('NOR','NormalMetal1',58000000,50,50);

       % Define a new resistor metal type named 'ResistorMetal1'
       % with a resistance of 100.
       Project.defineNewMetalType('RES','ResistorMetal1',100);

       % Define a new native metal type named 'NativeMetal1'
       % with a resistance of 100 and a skin coefficient of 50
       Project.defineNewMetalType('NAT','NativeMetal1',100,50);

       % Define a new general metal type named 'GeneralMetal1'
       % with a resistance of 100, a skin coefficient of 50,
       % a reactance of 50 and a kinetic inductance of 50.
       Project.defineNewMetalType('SUP','GeneralMetal1',100,50,50,50);

       % Define a new sense metal type named 'SenseMetal1'
       % with a reactance of 50
       Project.defineNewMetalType('SEN','SenseMetal1',50);

       % Define a new thick metal type named 'ThickMetal1'
       % with a conductivity of 58000000, a current ratio of 50,
       % a thickness of 50, and is comprised of 2 sheets.
       Project.defineNewMetalType('TMM','ThickMetal1',58000000,50,50,2);

       % Define a new rough metal type named 'RoughMetal1'
       % modeled as thick metal with a thickness of 5 units,
       % a conductivity of 58000000, a current ratio value of
       % zero and top/bottom roughness values of 1.1.
       Project.defineNewMetalType('RUF','RoughMetal1','thick',5,58000000,0,1.1,1.1);

   See also SonnetProject/defineNewNormalMetalType,
            SonnetProject/defineNewResistorMetalType,
            SonnetProject/defineNewNativeMetalType,
            SonnetProject/defineNewGeneralMetalType,
            SonnetProject/defineNewSenseMetalType,
            SonnetProject/defineNewThickMetalType,
            SonnetProject/defineNewRoughMetalType



defineNewNativeMetalType   Defines a new type of metal
   Project.defineNewNativeMetalType(Name,Resistance,SkinCoefficient) will
   add a native type of metal to the array of metals.

   Note: This method is only for geometry projects.

   Example usage:

       % Define a new native metal type named 'NativeMetal1'
```

```
        % with a resistance of 100 and a skin coefficient of 50
        Project.defineNewNativeMetalType('NativeMetal1',100,50);

    See also SonnetProject/defineNewMetalType



defineNewNormalMetalType   Defines a new type of metal
    Project.defineNewNormalMetalType(Name,Conductivity,CurrentRatio,Thickness)
    will add a normal type of metal to the array of metals.

    Note: This method is only for geometry projects.

    Example usage:

        % Define a new normal metal type named 'Copper'
        % of conductivity 58000000, current ratio 0 and thickness 1.4.
        Project.defineNewNormalMetalType('Copper',58000000,0,1.4);

    See also SonnetProject/defineNewMetalType



defineNewResistorMetalType   Defines a new type of metal
    Project.defineNewResistorMetalType(Name,Resistance) will
    add a resistor type of metal to the array of metals.

    Note: This method is only for geometry projects.

    Example usage:

        % Define a new resistor metal type named 'ResistorMetal1'
        % with a resistance of 100.
        Project.defineNewResistorMetalType('ResistorMetal1',100);

    See also SonnetProject/defineNewMetalType



defineNewRoughMetalType   Defines a new type of metal
    Project.defineNewRoughMetalType(...) will add a rough type of
    metal to the array of metal types.

    defineNewRoughMetalType requires the following arguments:
      1) The Name of the metal
      2) Whether the metal should be modeled as being thick or thin
           This value may be either (case insensitive)
               - 'thick' or 'THK' for thick
               - 'thin' or 'THN' for thin
      3) The Thickness of the metal
      4) The Conductivity of the metal
      5) The Current Ratio of the metal
      6) The Roughness of the top
      7) The Roughness of the bottom

    Note: This method is only for geometry projects.

    Example usage:

        % Define a new rough metal type named 'RoughMetal1'
        % modeled as thick metal with a thickness of 5 units,
        % a conductivity of 58000000, a current ratio value of
        % zero and top/bottom roughness values of 1.1.
        Project.defineNewRoughMetalType('RoughMetal1','thick',5,58000000,0,1.1,1.1);

    See also SonnetProject/defineNewMetalType
```

```
defineNewSenseMetalType   Defines a new type of metal
   Project.defineNewSenseMetalType(Name,Reactance) will
   add a Sense type of metal to the array of metals.

   Note: This method is only for geometry projects.

   Example usage:

       % Define a new sense metal type named 'SenseMetal1'
       % with a reactance of 50
       Project.defineNewSenseMetalType('SenseMetal1',50);

   See also SonnetProject/defineNewMetalType



defineNewSurfaceMetalType   Defines a new type of via metal
   Project.defineNewSurfaceMetalType(Name,Rdc,Rrf,Xdc) will define
   a surface metal type for the project.

   Note: This method is only for geometry projects.
   Note: This method is only for Sonnet version 13 projects.

   Example usage:

       % Define a new surface metal type named 'SurfaceMetal1'
       Project.defineNewSurfaceMetalType('SurfaceMetal1',5,5,5);

   See also SonnetProject/defineNewViaMetalType



defineNewThickMetalType   Defines a new type of metal
   Project.defineNewThickMetalType(Name,Conductivity,CurrentRatio,
   Thickness,NumSheets) will add a Thick Metal type of metal to
   the array of metals.

   Note: This method is only for geometry projects.

   Example usage:

       % Define a new thick metal type named 'ThickMetal1'
       % with a conductivity of 100, a current ratio of 50,
       % a thickness of 50, and is comprised of 2 sheets.
       Project.defineNewThickMetalType('ThickMetal1',100,50,50,2);

   See also SonnetProject/defineNewMetalType



defineNewViaMetalType   Create a new type of via metal
   Project.defineNewViaMetalType(...) will add
   a via metal type to the project.

   There are two ways to use defineNewViaMetalType. The user
   may define a metal type using a set of custom options or
   the user may define a type using a predefined property set
   from the Sonnet library.

   If defineNewViaMetalType is to import a metal type from the
   Sonnet library then the following arguments must be specified
     1) The name of the metal
     2) The metal's thickness

   If defineNewViaMetalType is to be used to define a custom
```

```
   metal type then the user must first specify the type of
   metal that is being defined and then specify the parameters
   for the metal type.

   defineNewViaMetalType requires a type as
   the first argument which should
   be one of the following:

       VOL   -   Volume Metal
       SFC   -   Surface Metal
       ARR   -   Array Metal

   Then you will need to supply the necessary
   arguments for each type as follows:

   VOL - Volume Metal
     1) The Name of the metal
     2) The Conductivity of the metal (inf for infinite)
     3) The Wall thickness (-1 or 'Solid' for solid)

   SFC - Surface Metal
     1) The Name of the metal
     2) The Rdc value
     3) The Rrf value
     4) The Xdc value

   ARR - Array Metal
     1) The Name of the metal
     2) The Conductivity value
     3) The Fill Factor

   Note: This method is only for geometry projects.
   Note: This method is only for Sonnet version 13 projects.

   Example usage:

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity and a wall thickness of 1.4 mils.
       Project.defineNewViaMetalType('VOL','Aluminum',3.72e7,1.4);

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity with a solid via wall.
       Project.defineNewViaMetalType('VOL','Aluminum2',3.72e7,-1);

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity with a solid via wall.
       Project.defineNewVolumeMetalType('Aluminum2',3.72e7,'Solid');

       % Define a new array metal type named 'ArrayMetal1'
       Project.defineNewViaMetalType('ARR','ArrayMetal1',50,100);

       % Define a new surface metal type named 'SurfaceMetal1'
       Project.defineNewViaMetalType('SFC','SurfaceMetal1',5,5,5);

   See also SonnetProject/defineNewMetalType



defineNewVolumeMetalType   Defines a new type of via metal
   Project.defineNewVolumeMetalType(theName,theConductivity,
   theWallThickness) will define a volume metal type for the project.
   If the wall should be solid then either pass -1 as the wall thickness
   or the string 'Solid' (case insensitive).

   Note: This method is only for geometry projects.
   Note: This method is only for Sonnet version 13 projects.
```

```
   Example usage:

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity and a wall thickness of 1.4 mils.
       Project.defineNewVolumeMetalType('Aluminum',3.72e7,1.4);

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity with a solid via wall.
       Project.defineNewVolumeMetalType('Aluminum2',3.72e7,-1);

       % Make an aluminum volume metal type with 3.72e7 s/m
       % conductivity with a solid via wall.
       Project.defineNewVolumeMetalType('Aluminum2',3.72e7,'Solid');

   See also SonnetProject/defineNewViaMetalType



defineVariable   Define a Geometry/Netlist Variable
   Project.defineVariable(Name,Value) When used with
   a geometry project will define a new geometry
   variable. When used with a netlist project defineVariable
   will define a new Netlist parameter.

   Project.defineVariable(Name,Value,Type) When used with
   a geometry project will define a new geometry variable of
   the specified type. When used with a netlist project
   the value of Type is ignored and defineVariable
   will define a new Netlist parameter.

   Project.defineVariable(Name,Value,Type,Description) This
   command will operate the same as above except the user
   may supply a description for the newly created variable.
   Descriptions are only stored for geometry projects.

   Type may be one of the following values:
     LNG      Length
     RES      Resistance
     CAP      Capacitance
     IND      Inductance
     FREQ     Frequency
     OPS      Ohms/sq
     SPM      Siemens/meter
     PHPM     picoHenries/meter
     RRF      Rrf
     NONE     Undefined

   If the specified variable or parameter already
   exists its value will be replaced.

   Example usage:
       Project.defineVariable('Z0',50)
       Project.defineVariable('Length',50,'LNG')



deleteAllElements   Delete all circuit elements
   Project.deleteAllElements() will delete
   all circuit elements from the project.

   Note: This method is only for netlist projects.



deleteComponent   Delete a component
   Project.deleteComponent(Id) will delete the component
   with the passed ID from the array of components.
```

```
   Project.deleteComponent(Component) will delete the
   passed component from the array of components.

   Project.deleteComponent(Name) will delete the component
   with the passed name from the array of components.

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the component with debug ID 12
       Project.deleteComponent(12);



deletePolygonUsingId   Delete a polygon
   Project.deleteComponentUsingId(Id) will delete the component
   with the passed ID from the array of components.

   Project.deleteComponentUsingId(Component) will delete the
   passed component from the array of components.

   Project.deleteComponent(Name) will delete the component
   with the passed name from the array of components.

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the component with debug ID 12
       Project.deleteComponentUsingId(12);



deleteComponentUsingIndex   Deletes a component from the project
   Project.deleteComponentUsingIndex(N) will delete the Nth component
   in the array of components

   Project.deleteComponentUsingIndex(Component) will delete the
   passed component from the array of components.

   Project.deleteComponent(Name) will delete the component
   with the passed name from the array of components.

   This operation can also be achieved with
       Project.GeometryBlock.ArrayOfComponents(N)=[];

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the 5th component in the array of components
       Project.deleteComponentUsingIndex(5);



deleteDuplicatePoints   Deletes duplicate polygon points
   Project.deleteDuplicatePoints() will remove any
   duplicate points for all polygons in the project.

   Note: This method is only for geometry projects.

 See also SonnetProject/deleteDuplicatePolygons
```

```
deleteDuplicatePolygons   Deletes duplicate polygons
   Project.deleteDuplicatePolygons() will search for duplicate polygons in the
   project and delete one of the duplicate occurrences such
   that there will no longer be a pair of duplicate polygons.

   Note: This method is only for geometry projects.

   Example usage:

       Project.deleteDuplicatePolygons();

 See also SonnetProject/findDuplicatePolygons



deleteLayer   Deletes a layer from the project
   Project.deleteLayer(N) will delete the Nth
   dielectric layer from the array of
   dielectric layers.

   This operation can also be achieved with
   Project.GeometryBlock.SonnetBox.ArrayOfDielectricLayers(N)=[];

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the 2nd layer in the array of layers
       Project.deletePolygon(5);



deleteNetworkElement   Delete a network element
   Project.deleteNetworkElement(N) will delete
   the Nth network.

   Project.deleteNetworkElement(aNetwork) will delete
   the passed network element from the project.

   Project.deleteNetworkElement(aNetworkName) will
   delete the network element in the project with
   the matching name.

   Note: This method is only for netlist projects.



deletePolygon   Delete a polygon
   Project.deletePolygon(Id) will delete the polygon
   with the passed ID from the array of polygons. If any ports,
   edge vias or parameters are connected to the polygon then
   they will be deleted as well.

   Project.deletePolygon(Polygon) will delete the
   passed polygon from the array of polygons. If any ports,
   edge vias or parameters are connected to the polygon then
   they will be deleted as well.

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the polygon with debug ID 12
       BooleanWasThePolygonDeleted=Project.deletePolygon(12);
```

```
deletePolygonUsingId   Delete a polygon
   Project.deletePolygonUsingId(Id) will delete the polygon
   with the passed ID from the array of polygons. If any ports,
   edge vias or parameters are connected to the polygon then
   they will be deleted as well.

   Project.deletePolygonUsingId(Polygon) will delete the
   passed polygon from the array of polygons. If any ports,
   edge vias or parameters are connected to the polygon then
   they will be deleted as well.

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the polygon with debug ID 12
       BooleanWasThePolygonDeleted=Project.deletePolygonUsingId(12);



deletePolygonUsingIndex   Deletes a polygon from the project
   Project.deletePolygonUsingIndex(N) will delete the Nth polygon
   in the array of polygons. If any ports, edge vias or parameters
   are connected to the polygon then they will be deleted as well.

   Project.deletePolygonUsingIndex(Polygon) will delete the
   passed polygon from the array of polygons. If any ports,
   edge vias or parameters are connected to the polygon then
   they will be deleted as well.

   This operation can also be achieved with
       Project.GeometryBlock.ArrayOfPolygons(N)=[];

   Note: This method is only for geometry projects.

   Example usage:

       % Delete the 5th polygon in the array of polygons
       Project.deletePolygonUsingIndex(5);



deletePort   Deletes a port
   Project.deletePort(N) will delete
   the port represented by the port
   number N from the project.

   Note: This method is only for geometry projects.

   Example usage:
       % Delete port number one from a project
       Project.deletePort(1);

     See also SonnetProject/deletePortUsingIndex



deletePortUsingIndex   Deletes a port
   Project.deletePortUsingIndex(N) will delete
   the Nth port in the array of ports.

   Note: This method is only for geometry projects.

   Example usage:
       % Delete the first port in a project
       Project.deletePortUsingIndex(1);
```

```
      See also SonnetProject/deletePort


detectAllOptimizationVariables   Adds all optimization variables
   Project.detectAllOptimizationVariables() will make an
   optimization variable entry for every dimensional parameter
   in the project. All of the optimization parameters will be
   disabled by default.

   See also SonnetProject/editOptimizationVariable


disableCurrentCalculations   Disable current calculations
   Project.disableCurrentCalculations will disable current
   density calculation for this project. This setting can
   be enabled with the 'enableCurrentCalculations()' function.

   Note: This method is only for geometry projects.

   See also SonnetProject/viewCurrents,
            SonnetProject/enableCurrentCalculations


displayPolygons   Displays polygon information
   Project.displayPolygons() will print out
   the index, ID, centroid point, mean point,
   type, level and metal type for all the
   polygons in the project.

   displayPolygons('Short') will print out
   the index, ID, centroid point, mean point,
   type, level and metal type for all the
   polygons in the project.

   displayPolygons('Long') will print
   all of the properties for all of
   the polygons in the project.

   Note: This method is only for geometry projects.

   See also SonnetProject/drawCircuit


draw2d   2D circuit diagram
      Project.draw2d(theLevelNumber) will create
   a new Matlab figure that will plot a 2D view of
   the specified metalization level of the circuit.

   Note: This method is only for geometry projects.
   Note: This method is provides the same functionality
           as SonnetProject.drawLayer

   See also SonnetProject/drawCircuit, SonnetProject/drawLayer,
            SonnetProject/draw3d


draw3d   3D circuit diagram
      n=Project.draw3d() will create a new Matlab figure
   that will plot a 3D view of the circuit. The
   Matlab figure number will be n.

   n=drawCircuit(n) will use the Matlab figure
```

```
   window number n to draw a 3D view of the circuit.

   Note: This method is only for geometry projects.
   Note: This method is provides the same functionality
           as SonnetProject.drawCircuit

   See also SonnetProject/drawCircuit, SonnetProject/drawLayer,
            SonnetProject/draw2d



drawCircuit   3D circuit diagram
      n=Project.drawCircuit() will create a new Matlab figure
   that will plot a 3D view of the circuit. The
   Matlab figure number will be n.

   n=drawCircuit(n) will use the Matlab figure
   window number n to draw a 3D view of the circuit.

   Note: This method is only for geometry projects.
   Note: This method is provides the same functionality
           as SonnetProject.draw3d

   See also SonnetProject/draw3d, SonnetProject/drawLayer,
            SonnetProject/draw2d



drawLayer   2D circuit diagram
      Project.drawLayer(theLevelNumber) will create
   a new Matlab figure that will plot a 2D view of
   the specified metalization level of the circuit.

   Note: This method is only for geometry projects.
   Note: This method is provides the same functionality
           as SonnetProject.draw2d

   See also SonnetProject/drawCircuit, SonnetProject/draw2d,
            SonnetProject/draw3d



duplicatePolygon      Makes a copy of a polygon and adds it to the project
   Polygon=Project.duplicatePolygon(ID) Makes a copy of the polygon with the
   passed ID value and adds the copy to the end of the array of polygons.
   The new polygon will have a unique ID. The new polygon will be returned.

   Polygon=Project.duplicatePolygon(Polygon) Makes a copy of the passed
   polygon and adds the copy to the end of the array of polygons.
   The new polygon will have a unique ID. The new polygon will be returned.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygon,
           SonnetProject/copyPolygonUsingId,
           SonnetProject/copyPolygonUsingIndex,
           SonnetProject/duplicatePolygonUsingId,
           SonnetProject/duplicatePolygonUsingIndex



duplicatePolygonUsingId      Makes a copy of a polygon and adds it to the project
   Polygon=Project.duplicatePolygonUsingId(ID) Makes a copy of the
   polygon with the passed ID value and adds the copy to the end of
   the array of polygons. The new polygon will have a unique ID. The
   new polygon will be returned.
```

```
   Polygon=Project.duplicatePolygonUsingId(Polygon) Makes a copy of
   the passed polygon and adds the copy to the end of the array of
   polygons. The new polygon will have a unique ID. The new polygon
   will be returned.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygon,
           SonnetProject/copyPolygonUsingId,
           SonnetProject/copyPolygonUsingIndex,
           SonnetProject/duplicatePolygon,
           SonnetProject/duplicatePolygonUsingIndex



duplicatePolygonUsingIndex      Makes a copy of a polygon and adds it to the project
   Polygon=Project.duplicatePolygonUsingIndex(N) Makes a copy of the Nth
   polygon in the array of polygons and adds the copy to the end of the
   array of polygons. The new polygon will have a unique debug ID. The
   new polygon will be returned.

   Polygon=Project.duplicatePolygonUsingIndex(Polygon) Makes a copy of the
   passed polygon and adds the copy to the end of the array of polygons.
   The new polygon will have a unique ID. The new polygon will be returned.

   Note: This method is only for geometry projects.

  See also SonnetProject/copyPolygon,
           SonnetProject/copyPolygonUsingId,
           SonnetProject/copyPolygonUsingIndex,
           SonnetProject/duplicatePolygon,
           SonnetProject/duplicatePolygonUsingId



editOptimizationVariable   Edit values for an optimization variable
   Project.editOptimizationVariable(...) will allow users to edit the
   parameters for an optimization.

   This function requires the following inputs:
       1) The name of the variable to be modified
       2) The minimum value for the variable
       3) The maximum value for the variable
       4) The step value with which we are sweeping
           from the minimum value to the maximum value.
       5) Either 'Y' to specify the variable
           is being used or 'n' to specify that the
           variable is not being used.

   Example usage:

       Project.editOptimizationVariable('dim',5,10,1,'Y')

   See also SonnetProject/detectAllOptimizationVariables



enableCurrentCalculations   Enable current calculations
   Project.enableCurrentCalculations() will enable current
   density calculation for this project. The project will
   need to be simulated before current density information
   will be available. Be aware that current density
   calculations can be time consuming.

   Note: This method is only for geometry projects.

   See also SonnetProject/viewCurrents,
```

```
              SonnetProject/disableCurrentCalculations


 estimateMemoryUsage   Estimate memory usage
    [megabytes subsections]=Project.estimateMemoryUsage() will save the
    Sonnet project and call Sonnet's built in memory estimator.
    The number of megabytes required for simulation and the number
    of subsections are returned. The project must contain
    analysis frequencies before this method may be used.

    [megabytes subsections]=Project.estimateMemoryUsage() will save the
    Sonnet project and call Sonnet's built in memory estimator.
    The number of megabytes required for simulation and the number
    of subsections are returned. The project must contain
    analysis frequencies before this method may be used.

    Sonnet will only estimate the memory usages for geometry projects.

    Note: This method will save the project to the hard drive. If
          there hasn't been a filename associated with this project
          an error will be thrown. A filename may be specified using
          the saveAs method (see "help SonnetProject.saveAs")

    Example usage:
        % Use the most recently installed version of Sonnet
        % to estimate memory and subsections.
        [MegaBytesOfMemory NumberOfSubsections]=Project.estimateMemoryUsage();

        % Use Sonnet version 12.52 to estimate memory and subsections.
        [MegaBytesOfMemory NumberOfSubsections]=Project.estimateMemoryUsage('C:\Program
Files\sonnet.12.52');

    See also SonnetProject/simulate



 exportCurrents   Exports current data
    Project.exportCurrents(...) will call Sonnet and export
    the current data for a region of a layout. This method
    will save and simulate the project first. Current
    calculations will be enabled for the project.

    There are two approaches to calling this method:
    The first approach is the pass the method a
    Sonnet current data request configuration file.

        Example: Project.exportCurrents(aRequestFile);

    The second approach to calling this method involves passing
    arguments that would specify the output settings such that
    the method will essentially build an output configuration
    file. The arguments are the following:

       1) Region - The region must be either a JXYLine object,
                   a JXYRectangle object or []. If the region
                   is [] then the currents for the entire layout
                   will be outputted.
       2) Type   -  The type must be either 'JX','JY','JXY' or
                   'PWR' (for heat flux)
       3) Ports -  The ports should be either a vector of JXYPort
                   objects or a matrix that stores the voltage and
                   phase values for each port. The user only has
                   to define values for ports that have non-zero
                   voltage or phase values. When using a matrix
                   the data must be formatted as follows:
                      [PortNumber, Voltage, Phase;
```

```
                    PortNumber, Voltage, Phase; ...]
      4) Frequency -  A vector specifying the desired frequency values.
                      Values should be specified in the same units as the project.
      5) (Optional) X Grid Size - This determines the X direction resolution
                                  of the exported data. The grid size is the
                                  separation between two data points. The first
                                  value in the series is half of the grid size.
                                  Ex: a value of two would provide data at the
                                  points 1,3,5,7... If the grid X size is
                                  unspecified then the cell size from the project
                                  will be utilized. If the X grid size is []
                                  then the the cell size from the project
                                  will be utilized.
      6) (Optional) Y Grid Size - This determines the Y direction resolution of
                                  the exported data. If the grid Y size
                                  is unspecified then the cell size from the
                                  project will be utilized. If the X grid size
                                  is [] then the the cell size from the project
                                  will be utilized.
      7) (Optional) Level -  Specifies what metallization  level(s)
                             should be outputted. The level should be [] if
                             all levels should be outputted. The level should
                             be a single number (Ex: 4) if only one level
                             should be outputted. If a range of levels
                             should be outputted then the level should be a
                             vector in the form of [startLevel, endLevel].
      8) (Optional) Complex - Should be either true or false. True indicates that
                              current data should be returned as complex numbers.

   If the user would like to specify values for
   parameters they may use the last two arguments.

      9)  ParameterName  -    Should be either a vertical vector of strings
                              (use strvcat) or a cell array of strings.

      10) ParameterValue -    Should be either a vector or a cell array of values
                              such that the Nth element of ParameterValue is
                              the value for the parameter specified by the Nth
                              element of ParameterName.

   Note: This method is only for geometry projects.
   Note: This method will only work for Sonnet version 13 and later.
           This method will look for Sonnet 13 installations and use the
           one with the latest install date.
   Note: This method will save the project to the hard drive. If
           there hasn't been a filename associated with this project
           an error will be thrown. A filename may be specified using
           the saveAs method (see "help SonnetProject.saveAs")
   Note: The X grid size and Y grid size values may be empty matricies ([]).
           This will cause the script to default to the project's cell size.
           This ability allows you to use default values for the grid size
           but still set non-default values for the metalization level and
           complex data fields.

   See also SonnetProject/viewCurrents,
            SonnetProject/enableCurrentCalculations,
            SonnetProject/disableCurrentCalculations,
            SonnetProject/exportPattern



exportPattern   Exports pattern data
   Project.exportPattern(...) will call Sonnet and export
   the pattern data for a region of a layout. This method
   will save and simulate the project first. Current
   calculations will be enabled for the project.
```

```
   The arguments are the following:
       1) The PhiAngleVec [start stop step] of Phi (azimuthal angle) in degs.
       2) The ThetaAngleVec [start stop step] of Theta ("elevation" angle) in degs.
       3) The List of Frequencies at which the pattern should be calculated.
       4) The port excitations/terminations.
           This should be a matrix with columns:
               [PortNumber Magnitude Phase(deg) Real(Z) Imag(Z) Inductance Capacitance]
               example: [1 1 0 50 0 0 0]
               which means: [Port 1, MAG=1, PHASE=0, R=50, X=0, L=0, C=0]

   Note: This method is only for geometry projects.
   Note: This method will only work for Sonnet version 13 and later.
           This method will look for Sonnet 13 installations and use the
           one with the latest install date.
   Note: This method will save the project to the hard drive. If
         there hasn't been a filename associated with this project
         an error will be thrown. A filename may be specified using
         the saveAs method (see "help SonnetProject.saveAs")

   Example:
     % The below command will export pattern data for
     %   - Theta Values from 0 to 85 in steps of 1
     %   - Phi Value from 0 to 360 in steps of 1
     %   - Frequency Values of 2.4 GHz
     %   - Port 1 excitation: MAG=1, PHASE=0, R=50, X=0, L=0, C=0
     aPatternData=Project.exportPattern([0 360 1], [0 85 1], 2.4, [1 1 0 50 0 0 0]);

   See also SonnetProject/exportCurrents,
            SonnetProject/enableCurrentCalculations



findComponentUsingId   Search for a component using its ID

   [index component]=Project.findComponentUsingId(name) accepts
   the Debug ID for a component and returns the component's
   index in the array of components and a reference to the
   component. If the supplied component is not in the array
   then [] is returned.

   [index component]=Project.findComponentUsingId(Id) accepts
   the Debug ID for a component and returns the component's
   index in the array of components and a reference to the
   component. If the supplied component is not in the array
   then [] is returned.

   Note: This method is only for geometry projects.

   Example usage:
       % Find the component's index and obtain a reference to it
       [ComponentIndex,ComponentObject]=Project.findComponentUsingId('R1');

       % Find the component's index and obtain a reference to it
       [ComponentIndex,ComponentObject]=Project.findComponentUsingId(5);

 See also SonnetProject/getComponent



findDuplicatePolygons   Finds duplicate polygons
   [Polygons Indices NumberOfMatches]=Project.findDuplicatePolygons() searches
   for duplicate polygons in the project. The polygon references to the
   duplicates are returned along with their indices.

   Note: This method is only for geometry projects.

   Example usage:
```

```
       [Polygons PolygonIndex NumberOfMatches]=Project.findDuplicatePolygons();

 See also SonnetProject/deleteDuplicatePolygons



findParameterIndex   Find parameter index
   [numIndex arrayOfIndex]=Project.findParameterIndex(name) returns the
   number of indices and an array of indices index of the parameter
   in a Sonnet Project based on its name.

   Note: This method is only for geometry projects.



findPolygonIndex   Search for a polygon index
   index=Project.findPolygonIndex(Polygon) will search for the
   index of a polygon in the array of polygons. If the polygon
   is not found in the polygon array then [] is returned.

   Note: This method is only for geometry projects.

   Example usage:

       % Find the index of a particular polygon
       index=Project.findPolygonIndex(polygon);

 See also SonnetProject/scalePolygon



findPolygonUsingCentroidX   Finds a polygon given its centroid
   [polygon ID index]=Project.findPolygonUsingCentroidX(X) finds
   polygons have an centroid x coordinate of 'X'.

   [polygon ID index]=Project.findPolygonUsingCentroidX(X,Layer) finds
   polygons have an centroid x coordinate of 'X' on the metallization
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingCentroidX(X,Layer,Size) finds
   polygons have an centroid x coordinate of 'X' and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer that have a
       % centroid X value of zero.
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingCentroidX(0);

 See also SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingCentroidY
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint



findPolygonUsingCentroidXY   Finds a polygon given its centroid
   [polygon ID index]=Project.findPolygonUsingCentroidXY(X,Y) finds
   polygons have an centroid coordinate of ('X','Y').

   [polygon ID index]=Project.findPolygonUsingCentroidXY(X,Y,Layer) finds
   polygons have an centroid coordinate of ('X','Y') on the metallization
```

```
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingCentroidXY(X,Y,Layer,Size) finds
   polygons have an centroid coordinate of ('X','Y') and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer
       % that have a centroid at (0,0)
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingCentroidXY(0,0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint



findPolygonUsingCentroidY   Finds a polygon given its centroid
   [polygon ID index]=Project.findPolygonUsingCentroidY(Y) finds
   polygons have an centroid y coordinate of 'Y'.

   [polygon ID index]=Project.findPolygonUsingCentroidY(Y,Layer) finds
   polygons have an centroid y coordinate of 'Y' on the metallization
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingCentroidY(Y,Layer,Size) finds
   polygons have an centroid y coordinate of 'Y' and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer that have a
       % centroid Y value of zero.
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingCentroidY(0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint



findPolygonUsingFunction   Finds a polygon using a function
   [polygon ID index]=Project.findPolygonUsingFunction(Function) finds
   polygons using a particular user specified function.

   The passed function is expected to receive an argument of
   type SonnetGeometryPolygon and return a Boolean. The
   function should return true if the polygon should be
   included in the returned results.

   Because the polygon gets sent to a user made function the
   passed function may modify the polygon whilst inside the
   passed function.

   Note: This method is only for geometry projects.
```

```
   Example usage:
       % This dummy function returns all polygons
       % that have an X Centroid greater than 50.
       function result=dummySearch(polygon)
           if polygon.CentroidXCoordinate > 50
               result=true;
           else
               result=false;
           end
       end

       % Find all polygons on any layer that have a
       % centroid X coordinate greater than 50
       [PolygonObject PolygonId IndexInArray]=...
       Project.findPolygonUsingFunction(@dummySearch);

 See also SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint



findPolygonUsingId   Search for a polygon using its ID
   [index polygon]=Project.findPolygonUsingId(Id) accepts
   the Debug ID for a polygon and returns the polygon's
   index in the array of polygons and a reference to the
   polygon. If the supplied polygon is not in the array
   then [] is returned.

   Note: This method is only for geometry projects.

   Example usage:

       % Find the polygon's index and obtain a reference to it
       [polygonIndex,polygonObject]=Project.findPolygonUsingId(12);

 See also SonnetProject/findPolygonIndex



findPolygonUsingMeanX   Finds a polygon given its mean
   [polygon ID index]=Project.findPolygonUsingMeanX(X) finds
   polygons have an mean x coordinate of 'X'.

   [polygon ID index]=Project.findPolygonUsingMeanX(X,Layer) finds
   polygons have an mean x coordinate of 'X' on the metallization
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingMeanX(X,Layer,Size) finds
   polygons have an mean x coordinate of 'X' and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer that have a
       % mean X value of zero.
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingMeanX(0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingCentroidXY,
```

```
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint


findPolygonUsingMeanXY   Finds a polygon given its mean
   [polygon ID index]=Project.findPolygonUsingMeanXY(X,Y) finds
   polygons have an mean coordinate of ('X','Y').

   [polygon ID index]=Project.findPolygonUsingMeanXY(X,Y,Layer) finds
   polygons have an mean coordinate of ('X','Y') on the metallization
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingMeanXY(X,Y,Layer,Size) finds
   polygons have an mean coordinate of ('X','Y') and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer that have a
       % mean at (0,0)
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingMeanXY(0,0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingPoint


findPolygonUsingMeanY   Finds a polygon given its mean
   [polygon ID index]=Project.findPolygonUsingMeanY(Y) finds
   polygons have an mean y coordinate of 'Y'.

   [polygon ID index]=Project.findPolygonUsingMeanY(Y,Layer) finds
   polygons have an mean y coordinate of 'Y' on the metallization
   layer specified by 'Layer'.

   [polygon ID index]=Project.findPolygonUsingMeanY(Y,Layer,Size) finds
   polygons have an mean y coordinate of 'Y' and a size of 'Size' on
   the metallization  layer specified by 'Layer'.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer that have a
       % mean Y value of zero.
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingMeanY(0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanXY,
          SonnetProject/findPolygonUsingPoint


findPolygonUsingPoint   Find a polygon that contains a particular coordinate pair
   [polygon ID index]=Project.findPolygonUsingPoint(X,Y) finds a
   polygon in the array of polygons given an X and Y coordinate pair
```

```
   that is within the polygon. This method returns a reference to the
   polygon object, the polygon's Debug Id, and the index for the polygon
   in the cell array of polygons. If the supplied point is within more
   than one polygon all of the polygons are returned.

   [polygon ID index]=Project.findPolygonUsingPoint(X,Y,Level) finds a
   polygon in the array of polygons given an X and Y coordinate pair
   that is within the polygon. Only polygons on the specified layer are
   checked. This method returns a reference to the polygon object, the
   polygon's Debug Id, and the index for the polygon in the cell array
   of polygons. If the supplied point is within more than one polygon
   all of the polygons are returned.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all polygons on any layer
       % encompass the point (0,0)
       [PolygonObject PolygonId IndexInArray]=Project.findPolygonUsingPoint(0,0);

 See also SonnetProject/findPolygonUsingCentroidX,
          SonnetProject/findPolygonUsingCentroidY,
          SonnetProject/findPolygonUsingCentroidXY,
          SonnetProject/findPolygonUsingMeanX,
          SonnetProject/findPolygonUsingMeanY,
          SonnetProject/findPolygonUsingMeanXY



findPort   Finds a port
   [Port PortNumber Index]=Project.findPort(N) will
   find the port with the port number N in the array
   of ports.

   Note: This method is only for geometry projects.

   See also SonnetProject/findPortUsingPoint



findPortUsingPoint   Find a port given an approximate location
   [Port PortNumber Index]=Project.findPortUsingPoint(X, Y) finds a port
   in the array of ports given an (X,Y) coordinate pair that is near the port.
   This method returns a reference to the port object, the port number,
   and the index for the port in the cell array of ports. If all the ports
   are beyond a certain distance from the location then an error will be
   thrown.

   [Port PortNumber Index]=Project.findPortUsingPoint(X, Y, Level) finds a port
   in the array of ports given an (X,Y) coordinate pair that is near the port.
   only ports on the specified level will be checked. This method returns a
   reference to the port object, the port number, and the index for the port

   in the cell array of ports. If all the ports are beyond a certain distance
   from the location then an error will be thrown.

   Note: This method is only for geometry projects.

   Example usage:

       % Find all ports on any layer that are near (0,120)
       [thePort thePortNumber theIndex]=Project.findPortUsingPoint(0,120);

 See also SonnetProject/findPortsInGroup
```

```
findPortUsingPoint   Find a port given an approximate location
   [Port PortNumber Index]=Project.findPortsInGroup(GroupName) finds
   all the ports in the specified group name.

   Note: This method is only for geometry projects.

 See also SonnetProject/findPortUsingPoint



findVariableIndex find the index of a variable in the SonnetProject
   index=Project.findVariableIndex(name) returns the index of a variable in a Sonnet
   Project (layout or netlist) based on its name.



flipPolygonX   Flips a polygon about its center X axis
   Project.flipPolygonX(Polygon) will flip
   the passed polygon over its X axis.

   Project.flipPolygonX(ID) will flip the polygon
   which has the passed ID over its X axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       aPolygon=Project.GeometryBlock.ArrayOfPolygons{1};
       Project.flipPolygonX(aPolygon)



flipPolygonXUsingId   Flips a polygon about its center X axis
   Project.flipPolygonXUsingId(ID) will flip the
   polygon which has the passed ID over its X axis.

   Project.flipPolygonXUsingId(Polygon) will
   flip the passed polygon over its X axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       aId=Project.GeometryBlock.ArrayOfPolygons{1}.DebugId;
       Project.flipPolygonXUsingId(aId)



flipPolygonXUsingIndex   Flips a polygon about its center X axis
   Project.flipPolygonXUsingId(N) will flip the Nth
   polygon in the array of polygons over its X axis.

   Project.flipPolygonXUsingId(Polygon) will
   flip the passed polygon over its X axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       Project.flipPolygonX(1)
```

```
flipPolygonY   Flips a polygon about its center Y axis
   Project.flipPolygonY(Polygon) will flip
   the passed polygon over its Y axis.

   Project.flipPolygonY(ID) will flip the polygon
   which has the passed ID over its Y axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       aPolygon=Project.GeometryBlock.ArrayOfPolygons{1};
       Project.flipPolygonY(aPolygon)



flipPolygonY   Flips a polygon about its center Y axis
   Project.flipPolygonYUsingId(ID) will flip the
   polygon which has the passed ID over its Y axis.

   Project.flipPolygonYUsingId(Polygon) will
   flip the passed polygon over its Y axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       aId=Project.GeometryBlock.ArrayOfPolygons{1}.DebugId;
       Project.flipPolygonY(aId)



flipPolygonYUsingIndex   Flips a polygon about its center Y axis
   Project.flipPolygonYUsingId(N) will flip the Nth
   polygon in the array of polygons over its Y axis.

   Project.flipPolygonYUsingId(Polygon) will
   flip the passed polygon over its Y axis.

   Note: This method is only for geometry projects.

   Example usage:

       % Flip the first polygon in the
       % array of polygons.
       Project.flipPolygonY(1)



generateUniqueId   Generate a unique debugId
   Id=Project.generateUniqueId() will very quickly find a
   debugId that is not being used by any other
   polygons in the project. Values are not sequential
   but are found quickly even with a large number
   of polygons.

   If the project has no polygons a debugId of one
   is always returned.

   This method is useful when manually creating polygons
```

```
   or when wanting to make sure that cloned polygons
   have unique debugId values.

   Note: This method is only for geometry projects.

 See also SonnetProject/assignUniqueDebugId,
          SonnetProject/generateUniqueId



getAllPolygonCentroids   Generates vectors for centroids and references
   [X Y Layers Sizes Polygons]=Project.getAllPolygonCentroids() will
   return a vector of all of the centroid X coordinates, the
   centroid Y coordinates, the layers, the sizes and polygon
   handles for all the polygons in a project.

   Note: This method is only for geometry projects.

 See also SonnetProject/findPolygonIndex



getAllPolygonIds   Generates vectors of IDs and references
   [IDs Polygons]=Project.getAllPolygonIds() will return a vector of all of the
   polygon ID's in a project and a cell array of a reference
   to each polygon such that IDs(n) is the ID for Polygons(n).

   Note: This method is only for geometry projects.

 See also SonnetProject/findPolygonIndex



getAllPolygonMeans   Generates vectors for means and references
   [X Y Layers Sizes Polygons]=Project.getAllPolygonMeans() will
   return a vector of all of the mean X coordinates, the
   mean Y coordinates, the layers, the sizes and polygon
   handles for all the polygons in a project.

   Note: This method is only for geometry projects.

 See also SonnetProject/findPolygonIndex



getAllPolygonIds   Generates vectors for coordinates and references
   [X Y Layers Sizes Polygons]=Project.getAllPolygonPoints() will
   return arrays of all of the polygon X coordinates and
   the polygon Y coordinates.

   Note: This method is only for geometry projects.

 See also SonnetProject/findPolygonIndex



getCapacitorComponents   Returns capacitor components
   components=Project.getCapacitorComponents() searches
   for capacitor components and returns a vector of
   component references.

   Note: This method is only for geometry projects.

   Example usage:
       components=Project.getCapacitorComponents()
       length(components)   % The number of returned components
```

```
getComponent   Returns a component in the project
   aComponent=Project.getComponent(N) will return the Nth component
   in the array of components. This operation can also be achieved with
       component=Project.GeometryBlock.ArrayOfComponents{N};

   aComponent=Project.getComponent(ComponentName) will return the component
   in the array of components with the specified name. If the component
   is not found then [] will be returned.

   aComponent=Project.getComponent() will return the last component
   in the array of components.

   Note: This method is only for geometry projects.

   Example usage:

       % Get the 5th component in the array of components
       component=Project.getComponent(5);



addDataFileComponent   Returns data file components
   components=Project.getDataFileComponents() searches
   for data file components and returns a vector of
   component references.

   Note: This method is only for geometry projects.

   Example usage:
       components=Project.getDataFileComponents()
       length(components)   % The number of returned components



getInductorComponents   Returns inductor components
   components=Project.getInductorComponents() searches
   for inductor components and returns a vector of
   component references.

   Note: This method is only for geometry projects.

   Example usage:
       components=Project.getInductorComponents()
       length(components)   % The number of returned components



getLayer   Returns polygon in the project
   layer=Project.getLayer(N) will return the Nth
   dielectric layer in the array of
   dielectric layers.

   This operation can also be achieved with
       layer=Project.GeometryBlock.SonnetBox.ArrayOfDielectricLayers{N};

   Note: This method is only for geometry projects.

   Example usage:

       % Get the 2nd dielectric layer in the project
       layer=Project.getLayer(2);



getLayerIndexes  Returns the list of dielectric layer names
```

```
   aValue=Project.getLayerIndexes() returns a vertically
   concatinated vector of layer indexes. If the project
   has thick metal types the sublevels will be included.

   This method is useful for determining the indexes
   of levels (and sublevels defined by thick metal
   types) for JXY data exporting.

   Note: This method is only for geometry projects.
   Note: This method is only supported on Sonnet version 13
   Note: This method will perform a save operation. If the
           project does not yet have an associated filename
           the save will not be successful.

   Example:
     % Export level numbers from a project
     % with a defined thick metal type.
     Project.getLayerIndexes()



getMetalType   Get a metal type
   metal=getMetalType(name) will search the project
   for the specified metal type based on its name.
   if the metal is not found an error is thrown.



getNetworkElements   Returns network in a project
   aNetwork=Project.getNetworkElements(N) will return a cell
   array of all the circuit elements in the
   Nth network of a netlist.

   Note: This method is only for netlist projects.

   Example usage:

       % Get the 5th network in a netlist
       network=Project.getNetworkElements(5);



getPolygon   Returns polygon in the project
   aPolygon=Project.getPolygon(N) will return the Nth polygon
   in the array of polygons.

   aPolygon=Project.getPolygon() will return the last polygon
   in the array of polygons.

   This operation can also be achieved with
       polygon=Project.GeometryBlock.ArrayOfPolygons{N};

   Note: This method is only for geometry projects.

   Example usage:

       % Get the 5th polygon in the array of polygons
       polygon=Project.getPolygon(5);



getResistorComponents   Returns resistor components
   components=Project.getResistorComponents() searches
   for resistor components and returns a vector of
   component references.

   Note: This method is only for geometry projects.
```

```
   Example usage:
       components=Project.getResistorComponents()
       length(components)   % The number of returned components



getVariableValue  Returns the value of a variable
   aValue=Project.getVariableValue(name) returns
   the value of the variable specified by name.
   if the variable does not exist [] is returned.
   This method supports both geometry and netlist
   variables.



 initialize   Initializes a Sonnet geometry project
   Project.Initialize() initializes a project to default
   values for a Sonnet geometry project.

 See also  SonnetProject/initializeNetlist,
           SonnetProject/initializeGeometry



 initializeGeometry   Initializes a Sonnet geometry project
   Project.initializeGeometry() initializes a project to the
   default values for a Sonnet geometry project.

 See also SonnetProject/initialize,
          SonnetProject/initializeNetlist



 initializeNetlist   Initializes a Sonnet netlist project
   Project.initializeNetlist() initializes a project
   to the default values for a Sonnet netlist project.

 See also  SonnetProject/initialize,
           SonnetProject/initializeGeometry



isGeometryProject   Checks project type
   Boolean=Project.isGeometryProject returns true if the
   project is a geometry project; it returns
   false if it is a netlist project.

   Example usage:
       if Project.isGeometryProject()
           ....
       end

 See also SonnetProject/isNetlistProject



isMetalTypeDefined   Checks if a metal type is defined
   isDefined=isMetalTypeDefined(name) will return true
   or false depending on if any metals in the project
   have the same name.



isNetlistProject   Checks project type
   Boolean=Project.isNetlistProject returns true if the
   project is a netlist project; it returns
```

```
   false if it is a geometry project.

   Example usage:
       if Project.isNetlistProject()
           ....
       end

 See also SonnetProject/isNetlistProject



modifyVariableValue   Modify Geometry/Netlist Variable Value
   Project.modifyVariableValue(Name,Value) When used with
   a geometry project will modify the value for the geometry
   variable with the passed name. If there are
   any parameters associated with the variable then the
   parameter's values will be updated to be consistent. If
   Project is a netlist project then the value for the
   netlist variable will be modified.

   If the user supplies the name for an invalid variable name then no
   action will take place. The name of the variable is
   case insensitive.

   Example usage:
       Project.modifyVariableValue('Length',1)



movePolygon   Moves a polygon to a new X and Y location
   Project.movePolygon(polygon,X,Y) will move a polygon such that its centroid
   will be at the desired location.

   Project.movePolygon(polygon,X,Y) will move the passed
   polygon such that its centroid will be at the desired location.

   Note: This method is only for geometry projects.

   Example usage:

       % Move a particular polygon such that
       % its centroid is at location (0,0)
       Project.movePolygon(polygon,0,0);

       % Move the polygon with debug ID 12
       % such that its centroid is at location (0,0)
       Project.movePolygon(12,0,0);

 See also SonnetProject/movePolygonExact, SonnetProject/movePolygonRelative



movePolygonExact   Moves a polygon to a new X and Y location
   Project.movePolygonExact(Polygon,X,Y) will move
   a polygon such that its centroid will be at the
   desired location.

   Project.movePolygonExact(ID,X,Y) will move a
   polygon such that its centroid will be at the
   desired location.

   Note: This method is only for geometry projects.

   Example usage:

       % Move a particular polygon such that
       % its centroid is at location (0,0)
```

```
       Project.movePolygonExact(polygon,0,0);

       % Move the polygon with debug ID 12
       % such that its centroid is at location (0,0)
       Project.movePolygonExact(12,0,0);

 See also SonnetProject/movePolygon, SonnetProject/movePolygonRelative



movePolygonExactUsingId   Moves a polygon to a new X and Y location
   Project.movePolygonExactUsingId(ID,X,Y) will move a
   polygon such that its centroid will be at the
   desired location.

   Project.movePolygonExactUsingId(Polygon,X,Y) will move
   the passed polygon such that its centroid will be at
   the desired location.

   Note: This method is only for geometry projects.

   Example usage:

       % Move the polygon with debug ID 12
       % such that its centroid is at location (0,0)
       Project.movePolygonExactUsingId(12,0,0);

       % Move a particular polygon such that
       % its centroid is at location (0,0)
       Project.movePolygonExactUsingId(polygon,0,0);

 See also SonnetProject/movePolygon, SonnetProject/movePolygonRelative



movePolygonExactUsingIndex   Moves a polygon to a new X and Y location
   Project.movePolygonExactUsingIndex(N,X,Y) will move
   the Nth polygon in the array of polygons such that
   its centroid will be at the desired location.

   Project.movePolygonExactUsingIndex(Polygon,X,Y) will move
   the passed polygon such that its centroid will be at
   the desired location.

   Note: This method is only for geometry projects.

   Example usage:

       % Move a particular polygon such that
       % its centroid is at location (0,0)
       Project.movePolygonExactUsingIndex(polygon,0,0);

       % Move the polygon with debug ID 12
       % such that its centroid is at location (0,0)
       Project.movePolygonExactUsingIndex(12,0,0);

 See also SonnetProject/movePolygon, SonnetProject/movePolygonRelative



movePolygonRelative   Moves a polygon by a particular amount
   Project.movePolygonRelative(Polygon,X,Y) will move a
   polygon such that its centroid X value will be moved
   by the specified distance for the X direction and
   the centroid Y value will be moved by the specified
   distance in the Y direction.
```

```
   Project.movePolygonRelative(ID,X,Y) will move the polygon
   with the passed ID such that its centroid X value will be
   moved by the specified distance for the X direction
   and the centroid Y value will be moved by the specified
   distance in the Y direction.

   Note: This method is only for geometry projects.

   Example usage:

       % Move a particular polygon such that
       % its centroid is at location (0,0)
       Project.movePolygonRelative(polygon,0,0);

       % Move the polygon with debugID 12
       % such that its centroid is at location (0,0)
       Project.movePolygonRelative(12,0,0);

 See also SonnetProject/movePolygon, SonnetProject/movePolygonExact



movePolygonRelativeUsingId   Moves a polygon by a particular amount
   Project.movePolygonRelativeUsingId(ID,X,Y) will move the polygon
   with the passed ID such that its centroid X value will be
   moved by the specified distance for the X direction
   and the centroid Y value will be moved by the specified
   distance in the Y direction.

   Project.movePolygonRelativeUsingId(Polygon,X,Y) will move a
   polygon such that its centroid X value will be moved
   by the specified distance for the X direction and
   the centroid Y value will be moved by the specified
   distance in the Y direction.

   Note: This method is only for geometry projects.

   Example usage:

       % Move the polygon with debugID 12
       % such that its centroid is at location (0,0)
       Project.movePolygonRelative(12,0,0);

       % Move a particular polygon such that
       % its centroid is at location (0,0)
       Project.movePolygonRelative(polygon,0,0);

 See also SonnetProject/movePolygon, SonnetProject/movePolygonExact



movePolygonRelativeUsingIndex   Moves a polygon by a particular amount
   Project.movePolygonRelativeUsingId(N,X,Y) will move the Nth polygon
   in the array of polygons such that its centroid X value will be
   moved by the specified distance for the X direction
   and the centroid Y value will be moved by the specified
   distance in the Y direction.

   Project.movePolygonRelativeUsingId(Polygon,X,Y) will move a
   polygon such that its centroid X value will be moved
   by the specified distance for the X direction and
   the centroid Y value will be moved by the specified
   distance in the Y direction.

   Note: This method is only for geometry projects.

   Example usage:
```

```
        % Move a particular polygon such that
        % its centroid is at location (0,0)
        Project.movePolygonRelativeUsingIndex(polygon,0,0);

        % Move the polygon with debugID 12
        % such that its centroid is at location (0,0)
        Project.movePolygonRelativeUsingIndex(12,0,0);
```

See also SonnetProject/movePolygon, SonnetProject/movePolygonExact

movePolygon Moves a polygon to a new X and Y location

Project.movePolygonUsingId(ID,X,Y) will move the polygon specified by the passed ID value such that its centroid will be at the desired location.

Project.movePolygonUsingId(polygon,X,Y) will move the passed polygon such that its centroid will be at the desired location.

Note: This method is only for geometry projects.

Example usage:

```
        % Move the polygon with debug ID 12
        % such that its centroid is at location (0,0)
        Project.movePolygonUsingId(12,0,0);

        % Move a particular polygon such that
        % its centroid is at location (0,0)
        Project.movePolygonUsingId(polygon,0,0);
```

See also SonnetProject/movePolygonExact, SonnetProject/movePolygonRelative

movePolygon Moves a polygon to a new X and Y location

Project.movePolygonUsingIndex(N,X,Y) will move the Nth polygon in the array of polygons such that its centroid will be at the desired location.

Project.movePolygonUsingIndex(polygon,X,Y) will move the passed polygon such that its centroid will be at the desired location.

Note: This method is only for geometry projects.

Example usage:

```
        % Move a particular polygon such that
        % its centroid is at location (0,0)
        Project.movePolygonUsingIndex(polygon,0,0);

        % Move the polygon with index 3
        % such that its centroid is at location (0,0)
        Project.movePolygonUsingIndex(3,0,0);
```

See also SonnetProject/movePolygonExact, SonnetProject/movePolygonRelative

openInGui Opens a project in the Matlab GUI

Project.openInGui() saves the project and opens it in the GUI bundled with this interface. The user may edit the project in the GUI interface as much as they like. Once the user is done they can close the GUI interface and the project will re-update to reflect the changes made by the GUI.

```
   openInGui(Boolean) takes an argument to specify
   whether or not execution of the function should
   halt when the GUI window is open.  If the argument is
   a Boolean true the function will operate under its normal
   behavior and launch the GUI, wait for the GUI to be
   closed and update the changes to the project.  If the
   argument is a Boolean false the GUI window will be
   launched but the execution state will continue and
   the project changes will not be saved to the
   Sonnet project object that exists in memory (although
   changes may be saved to the version that exists on
   the hard drive if the save button in the GUI is pressed).

   Note: This method is only for geometry projects.
   Note: This method requires that the optional Matlab
         GUI be included in Matlab's path. The Matlab
         GUI is available from the Matlab Central File
         Exchange.

   Example usage:

       % Open the GUI and wait for GUI to be closed and
       % update the project settings.
       aSonnetProject.openInGui();
           % Or
       aSonnetProject.openInGui(true);

       % Open the GUI and do not wait for GUI to be closed.
       % and project settings will not be updated in Matlab.
       aSonnetProject.openInGui(false);

   See also SonnetProject/openInSonnet



openInSonnet   Opens a project in the Sonnet GUI
   Project.openInSonnet() saves the project and opens it in Sonnet. The
   user can then edit the project in Sonnet. Once the user
   is done they can close Sonnet and the version of the project
   that exists in Matlab will be updated to reflect the changes
   made in Sonnet.

   openInSonnet(Boolean) takes an argument to specify
   whether or not execution of the function should
   halt when the Sonnet window is open.  If the argument is
   a Boolean true the function will operate under its normal
   behavior and launch Sonnet, wait for Sonnet to be
   closed and update the changes to the project.  If the
   argument is a Boolean false the Sonnet window will be
   launched but the execution state will continue and
   the project changes will not be saved to the
   Sonnet project object that exists in memory (although
   changes may be saved to the version that exists on
   the hard drive if the save button in the Sonnet is pressed).

   openInSonnet(Boolean,Path) takes an argument to specify
   whether or not execution of the function should
   halt when the Sonnet window is open.  If the argument is
   a Boolean true the function will operate under its normal
   behavior and launch Sonnet, wait for Sonnet to be
   closed and update the changes to the project.  If the
   argument is a Boolean false the Sonnet window will be
   launched but the execution state will continue and
   the project changes will not be saved to the
   Sonnet project object that exists in memory (although
   changes may be saved to the version that exists on
```

```
   the hard drive if the save button in the Sonnet is pressed).
   The Path value specifies the directory which has the
   version of Sonnet that should be used.

   Example usage:

       % Opens the project with Sonnet and waits for Sonnet to be
       % closed. The project's settings will not be updated in Matlab.
       aSonnetProject.openInSonnet();
           % Or
       aSonnetProject.openInSonnet(true);

       % Opens the project with Sonnet and does not wait for Sonnet to be
       % closed. The project's settings will not be updated in Matlab.
       aSonnetProject.openInSonnet(false);

       % Opens the project with Sonnet version 12.52. This call will not
       % wait for Sonnet to be closed and the project's settings will not
       % be updated in Matlab.
       aSonnetProject.openInSonnet(false,'C:\Program Files\sonnet.12.52');

   See also SonnetProject/openInGui



polygonCount   Counts project's polygons
   n=Project.polygonCount()  Will return the number of
   polygons in the project.

   This operation can also be achieved with
       length(Project.GeometryBlock.ArrayOfPolygons)

   Note: This method is only for geometry projects.

   Example usage:

       % Get the number of polygons
       n=Project.polygonCount();



 quickClone   Initializes a replica project
   newProject=Project.quickClone() will return a deep
   copy of a Sonnet project. The copy will have all the
   same values for the class properties but will contain
   completely separate handles.

   The new project will have no filename associated
   with it but it may be saved with the saveAs()
   command.

   This method is typically much faster than clone()
   but requires a disk operation.

   Example usage:

       % Create a new Sonnet project object
       Project1=SonnetProject('project.son');

       % Clone the project
       Project2=Project1.quickClone();

       % Any modifications made to Project1
       % or Project2 will not affect the
       % other project.

 See also  SonnetProject/clone
```

```
removeAllDielectricBricks   Removes all bricks
   Project.removeAllDielectricBricks() will look through the
   array of polygons and delete any dielectric brick polygons.

   Note: This method is only for geometry projects.

   This function is useful if you are using dielectric
   bricks as a placeholder for objects.



removeOption   Removes values from option string
   removeOption(str) will removed the specified
     text from the project option string.



replaceDielectricLayer   Replace an existing dielectric layer
   Project.replaceDielectricLayer(...) will replace an existing
   dielectric layer in the stackup.

   There are two ways to use replaceDielectricLayer. The user
   may define a layer using a set of custom options or
   the user may define a using a predefined property set
   from the Sonnet library.

   Users may use replaceDielectricLayer to replace a layer with an isotropic dielectric
   layer in the project using the following parameters:
       1)  The array position for the layer to be replaced
       2)  Name of the Dielectric Layer
       3)  Thickness of the layer
       4)  Relative Dielectric Constant
       5)  Relative Magnetic Permeability
       6)  Dielectric Loss Tangent
       7)  Magnetic Loss Tangent
       8)  Dielectric Conductivity

   The user may also complete the same operation with an anisotropic
   layer by using the following parameters:
       1)  The array position for the layer to be replaced
       2)  Name of the Dielectric Layer
       3)  Thickness of the layer
       4)  Relative Dielectric Constant
       5)  Relative Magnetic Permeability
       6)  Dielectric Loss Tangent
       7)  Magnetic Loss Tangent
       8)  Dielectric Conductivity
       9)  Relative Dielectric Constant for Z Direction
       10)  Relative Magnetic Permeability for Z Direction
       11) Dielectric Loss Tangent for Z Direction
       12) Magnetic Loss Tangent for Z Direction
       13) Dielectric Conductivity for Z Direction

   Users may replace an existing layer with one based
   on an entry from the Sonnet library by using the
   following parameters:
       1) The array position for the layer to be replaced
       2) The name of the material (Ex: "Rogers RT6006")
       3) Thickness of the layer

   If no dielectric layer exists in the SonnetLibrary
   with the specified name then an error will be thrown.

   Note: This method is only for geometry projects.
```

```
   Example usage:
       % Replace the second dielectric layer with a layer which
       % is 10 units thick, has a relative dielectric constant
       % of 1, a relative magnetic permeability of 1,
       % a dielectric loss tangent of 0, a magnetic loss
       % tangent of 0, an dielectric conductivity of 0.
       Project.replaceDielectricLayer(2,'newLayer',10,1,1,0,0,0);

       % Replace the third layer of the project with an anisotropic
       % dielectric layer. The new layer is 10 units thick, has a
       % relative dielectric constant of 1, a relative magnetic
       % permeability of 1, a dielectric loss tangent of 0, a
       % magnetic loss tangent of 0, an dielectric conductivity of 0.
       % The Z direction has a relative dielectric constant
       % of 1, a dielectric loss tangent of 1, a magnetic
       % loss tangent of 0, and an dielectric conductivity of 0.
       Project.replaceDielectricLayer(3,'newLayer',10,1,1,0,0,0,1,1,0,0,0);

       % Replace the first layer's material with Rogers RT6006
       Project.replaceDielectricLayer(1,'Rogers RT6006',50);

 See also SonnetProject/addAnisotropicDielectricLayer



returnSelectedFrequencySweep   Returns a reference to the selected frequency sweep
   [sweep index]=Project.returnSelectedFrequencySweep() will
   return a handle to the object for the selected
   frequency sweep and its location in the
   array of frequency sweeps.

   If the frequency sweep type was combination
   then the return values will be a cell array of
   frequency sweep objects and a vector of list
   indices.

   This function cannot be used when the selected
   frequency sweep is parameter sweep, optimize or
   external file.



save   Saves a project to the hard drive
   Project.save() writes the project to a file.
   The file will be saved to the same filename
   as was used by the most recent call to saveAs. If
   saveAs has never been called it will use the name of
   the file that was originally opened by SonnetProject.
   If the project was made from scratch and has never
   been saved with savesAs then an error will be thrown.

   Example usage:
       Project.save();

 See also SonnetProject/saveAs



saveAs   Saves a project to the hard drive
   Project.saveAs(Filename) writes the Sonnet project to a
      file on the hard drive with the specified filename. If
   the operation involves overwritting a pre-existing project
   file the old project's simulation data will be deleted.

   Project.saveAs(Filename,clean) writes the Sonnet project
      to a file on the hard drive with the specified filename.
```

```
   If the clean argument is a boolean true then any preexisting
   simulation data will be removed.

   Note: This function will change the internal filename
   property for the project such that future calls to save()
   will save to this filename rather than the original filename.

   Note: Be careful when using the optional argument to not
   clear project data. Simulation results from the overwritten
   project may not be consistant with the new project and may
   provide incorrect simulation results.

   Example usage:

       % Save the project as 'project.son'
       Project.saveAs('project.son');

       % Save the project as 'project2.son' and will not
       % delete simulation data which existed for the old
       % version of the project.
       Project.saveAs('project.son');

 See also SonnetProject/save



scalePolygon   Expands polygons
   Project.scalePolygon(Polygon,XChange,YChange) will
   increase the size of a polygon by multiplying all of its coordinates
   by the specified X change factor and Y change factor. The polygon
   is scaled from its centroid so the polygon's position does not change.

   Project.scalePolygon(ID,XChange,YChange) will increase
   the size of the polygon with the passed ID by multiplying all of
   its coordinates  by the specified X change factor and Y change factor.
   The polygon is scaled from its centroid so the polygon's position does
   not change.

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygon(polygon,2,2);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygon(12,2,2);

 See also SonnetProject/scalePolygonFromPoint



scalePolygonFromPoint   Expands polygons
   scalePolygonFromPoint(Polygon,X,Y) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the centroid. This provides the
   same functionality as scalePolygon().

   scalePolygonFromPoint(Polygon,X,Y,PX,PY) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the coordinate location (PX,PY).

   scalePolygonFromPoint(ID,X,Y) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the centroid. This provides the
```

```
   same functionality as scalePolygon().

   scalePolygonFromPoint(ID,X,Y,PX,PY) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the coordinate location (PX,PY).

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPoint(polygon,2,2,0,0);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPoint(12,2,2,0,0);

 See also SonnetProject/scalePolygon



scalePolygonFromPointUsingId   Expands polygons
   scalePolygonFromPointUsingId(ID,X,Y) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the centroid. This provides the
   same functionality as scalePolygon().

   scalePolygonFromPointUsingId(ID,X,Y,PX,PY) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the coordinate location (PX,PY).

   scalePolygonFromPointUsingId(Polygon,X,Y) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the centroid. This provides the
   same functionality as scalePolygon().

   scalePolygonFromPointUsingId(Polygon,X,Y,PX,PY) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the coordinate location (PX,PY).

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPointUsingId(polygon,2,2,0,0);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPointUsingId(12,2,2,0,0);

 See also SonnetProject/scalePolygon



scalePolygonFromPointUsingIndex   Expands polygons
   scalePolygonFromPointUsingIndex(N,X,Y) will increase the size
   of the Nth polygon in the array of polygons by scaling
   the polygon by factors in the X and Y directions with
   respect to the centroid. This provides the
   same functionality as scalePolygon().
```

```
   scalePolygonFromPointUsingIndex(N,X,Y,PX,PY) will increase the size
   of the Nth polygon in the array of polygons by scaling the
   polygon by factors in the X and Y directions with respect
   to the coordinate location (PX,PY).

   scalePolygonFromPointUsingIndex(Polygon,X,Y) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the centroid. This provides the
   same functionality as scalePolygon().

   scalePolygonFromPointUsingIndex(Polygon,X,Y,PX,PY) will increase the size
   of a polygon by scaling the polygon by factors in the X and Y
   directions with respect to the coordinate location (PX,PY).

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPointUsingIndex(polygon,2,2,0,0);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       % with respect to the point (0,0)
       Project.scalePolygonFromPointUsingIndex(12,2,2,0,0);

 See also SonnetProject/scalePolygon



scalePolygonUsingId   Expands polygons
   Project.scalePolygonUsingId(ID,XChange,YChange) will increase
   the size of the polygon with the passed ID by multiplying all of
   its coordinates  by the specified X change factor and Y change factor.
   The polygon is scaled from its centroid so the polygon's position does
   not change.

   Project.scalePolygonUsingId(Polygon,XChange,YChange) will
   increase the size of a polygon by multiplying all of its coordinates
   by the specified X change factor and Y change factor. The polygon
   is scaled from its centroid so the polygon's position does not change.

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygonUsingId(12,2,2);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygonUsingId(polygon,2,2);

 See also SonnetProject/scalePolygonFromPoint



scalePolygonUsingIndex   Expands polygons
   Project.scalePolygonUsingIndex(N,XChange,YChange) will increase
   the size of the Nth polygon in the array of polygons by multiplying all of
   its coordinates  by the specified X change factor and Y change factor.
   The polygon is scaled from its centroid so the polygon's position does
   not change.
```

```
   Project.scalePolygonUsingIndex(Polygon,XChange,YChange) will
   increase the size of a polygon by multiplying all of its coordinates
   by the specified X change factor and Y change factor. The polygon
   is scaled from its centroid so the polygon's position does not change.

   Note: This method is only for geometry projects.

   Example usage:

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygonUsingIndex(polygon,2,2);

       % Scale a particular polygon such that
       % it is twice as large in the X and Y directions
       Project.scalePolygonUsingIndex(12,2,2);

 See also SonnetProject/scalePolygonFromPoint



setNetworkPorts   Sets the ports for a network
   Project.setNetworkPorts(N,[1 2 3 ...]) will modify the
   ports for the Nth network in the project to be the
   numbers specified in the second argument.

   Project.setNetworkPorts([1 2 3 ...]) will modify the
   ports for the last network in the project to be the
   numbers specified in the second argument. The last
   network in a project is the main network and
   specifies the external ports.

   Note: This method is only for netlist projects.

   Example usage:

       % Modify the ports for the fifth
       % network to be one through five.
       Project.setNetworkPorts(5,[1 2 3 4 5]);

       % Modify the ports for the last
       % network to be one through ten.
       Project.setNetworkPorts(1:10);



setOptions   Adds values to option string
   setOptions(str) will replace the project options
   test with the specified text. All previously
   defined options are lost.



simulate   Simulates Sonnet projects
   [success message]=Project.simulate() saves the project and calls
   Sonnet em to simulate the Sonnet Project File. If the simulation
   is successful then 'success' will be true; otherwise it
   will be false. Error messages returned from em will
   be stored in 'message'.

   [success message]=Project.simulate(Options) saves the project and
   calls Sonnet em to simulate the project with some particular options
   as defined below. If the simulation is successful then 'success' will
   be true; otherwise it will be false. Error messages returned from em
   will be stored in 'message'.
```

```
   Options are passed as a single
   string. Order of option switches does not
   matter and unknown option switches are
   ignored.

   Supported option switches:
       '-c'             To clean the project data first
       '-x'             To not clean the project data first (default)
       '-w'             To display a simulation status window (default)
       '-t'             To not display a simulation status window
       '-r'             To run the simulation instantaneously (default)
       '-p'             To not run the simulation instantaneously (requires status window)
       '-v' <VERSION>   To use a particular version of Sonnet to do the simulation (PC only)
       '-s' <DIRECTORY> To manually specify the Sonnet directory to
                        use for the simulation. The directory may either
                        be the base Sonnet directory or the version's bin
                        directory.

   Note: This method will save the project to the hard drive. If
         there hasn't been a filename associated with this project
         an error will be thrown. A filename may be specified using
         the saveAs method (see "help SonnetProject.saveAs")

   Example usage:

       % The project is written to a file and
       % simulated using the GUI status window
       aSonnetProject.simulate();

       % The project is written to a file and
       % simulated without displaying the status window
       aSonnetProject.simulate('-t');

       % The project is written to a file, cleaned
       % and then simulated without a GUI status window
       aSonnetProject.simulate('-t -c');

       % The project is simulated using the version of Sonnet
       % that exists in the specified location.
       aSonnetProject.simulate('-s C:\Program Files\sonnet.12.56'); % PC
       aSonnetProject.simulate('-s /disku/app/sonnet/13.54'); % Unix

   See also SonnetProject/estimateMemoryUsage, SonnetProject/viewResponseData,
            SonnetProject/viewCurrents



snapPolygonsToGrid   Snaps polygons to the grid
   Project.snapPolygonsToGrid() will snap all the
   polygons in a project to the grid in both the
   X and Y directions.

   Project.snapPolygonsToGrid(axis) will snap all
   polygons to the grid in the direction(s) specified
   by axis. snapPolygonsToGrid(axis) will call the
   appropriate snap method to either snap to the X
   axis, the Y axis or both.

   The user can specify the axis with one of the following strings:
           'x'  or 'X'  for the X direction
           'Y'  or 'Y'  for the X direction
           'xy' or 'XY' for the X and Y directions

           If an invalid axis string is supplied an 'XY' snap will be performed.

   Note: This method is only for geometry projects.
```

```
   Example usage:

       % Snap polygons in the X direction
       Project.snapPolygonsToGrid('x');

       % Snap polygons in the X and Y directions
       Project.snapPolygonsToGrid();
           % or
       Project.snapPolygonsToGrid('XY');



stringSignature   Returns the project file as a string
   string=Project.stringSignature() returns a string which would
   contain all the information that would normally
   be present when saving a project to the disk.

   Example usage:
       aString=Project.stringSignature();

 See also SonnetProject/save



symmetryOff   Turns symmetry off for the project
   Project.symmetryOff() Will turn symmetry off for
   the top and bottom halves of the project layout.

   Note: This method is only for geometry projects.

   Example usage:
       Project.symmetryOff();

   See also SonnetProject/symmetryOn



symmetryOn   Turns symmetry on for the project
   Project.symmetryOn() Will turn symmetry on for
   the top and bottom halves of the project layout.

   Note: This method is only for geometry projects.

   Example usage:
       Project.symmetryOn();

   See also SonnetProject/symmetryOff



viewCurrents   Launches current viewer
   Project.viewCurrents() will call Sonnet's built in
   current density viewer application to view the currents
   for the project. The project must have had the compute
   current setting on in order for the currents to have been
   calculated while simulating.  This can be enabled using
   the 'enableCurrentCalculations()' function.

   Project.viewCurrents(Path) will call Sonnet's built in
   current density viewer application to view the currents
   for the project. The method will use the version of Sonnet
   located at the specified directory. The project must have
   had the compute current setting on in order for the currents
   to have been calculated while simulating.  This can be
   enabled using the 'enableCurrentCalculations()' function.

   Note: This method is only for geometry projects.
```

```
   Example:
       % View currents using the default version of Sonnet
       viewCurrents();

       % View currents using a particular version of Sonnet
       viewCurrents('C:\Program Files\sonnet.12.52')

   See also SonnetProject/viewResponseData,
            SonnetProject/enableCurrentCalculations,
            SonnetProject/disableCurrentCalculations



viewResponseData   Launches emgraph
   Project.viewResponseData() will open the project's response data
   using Sonnet's built in response analysis tool emgraph.
   The project must be simulated before viewing response files.

   Project.viewResponseData(Path) will open the project's response data
   using Sonnet's built in response analysis tool emgraph. The method
   will use the version of Sonnet located at the specified directory.
   The project must be simulated before viewing response files.

   Example:
       % View response using the default version of Sonnet
       viewResponseData();

       % View response using a particular version of Sonnet
       viewResponseData('C:\Program Files\sonnet.12.52')



xBoxSize   Return box size for X direction
   BoxSize=Project.xBoxSize() returns the total width of the Sonnet box.
   The Sonnet box is the rectangular area that represents the
   boundaries for a circuit.

   Note: This method is only for geometry projects.

   Example usage:

       % Get the cell size in the X direction
       number=Project.xBoxSize();

   See also SonnetProject/xCellSize, SonnetProject/yCellSize
            SonnetProject/yBoxSize



xCellSize   Return cell size for X direction
   CellSize=Project.xCellSize() determines the width of each cell in the grid.
   The grid is clearly visible in the Sonnet GUI. Polygons
   edges are typically along grid lines.

   Note: This method is only for geometry projects.

   Example usage:

       % Get the cell size in the X direction
       number=Project.xCellSize();

   See also SonnetProject/yCellSize, SonnetProject/xBoxSize
            SonnetProject/yBoxSize
```

```
yBoxSize   Return box size for Y direction
   BoxSize=Project.yBoxSize() returns the total height of the Sonnet box.
   The Sonnet box is the rectangular area that represents the
   boundaries for a circuit.

   Note: This method is only for geometry projects.

   Example usage:

       % Get the cell size in the Y direction
       number=Project.yBoxSize();

   See also SonnetProject/xCellSize, SonnetProject/yCellSize
            SonnetProject/yBoxSize



yCellSize   Return cell size for Y direction
   CellSize=Project.yCellSize() determines the height of each cell in the grid.
   The grid is clearly visible in the Sonnet GUI. Polygons
   edges are typically along grid lines.

   Note: This method is only for geometry projects.

   Example usage:

       % Get the cell size in the Y direction
       number=Project.yCellSize();

   See also SonnetProject/xCellSize, SonnetProject/xBoxSize
            SonnetProject/yBoxSize
```

Contact

Your feedback is important to us. If you have any questions or comments about SonnetLab, please contact Sonnet Support by email at support@sonnetsoftware.com.

Please make sure you are using the most up to date version of SonnetLab before submitting a bug report. When submitting a bug report please include the Sonnet project file that generated the error (Sonnet project files have the extension .son) and the output from the command "SonnetMatlabVersion". The more information that that we receive the faster it will be for us to resolve the issue and contact you back.