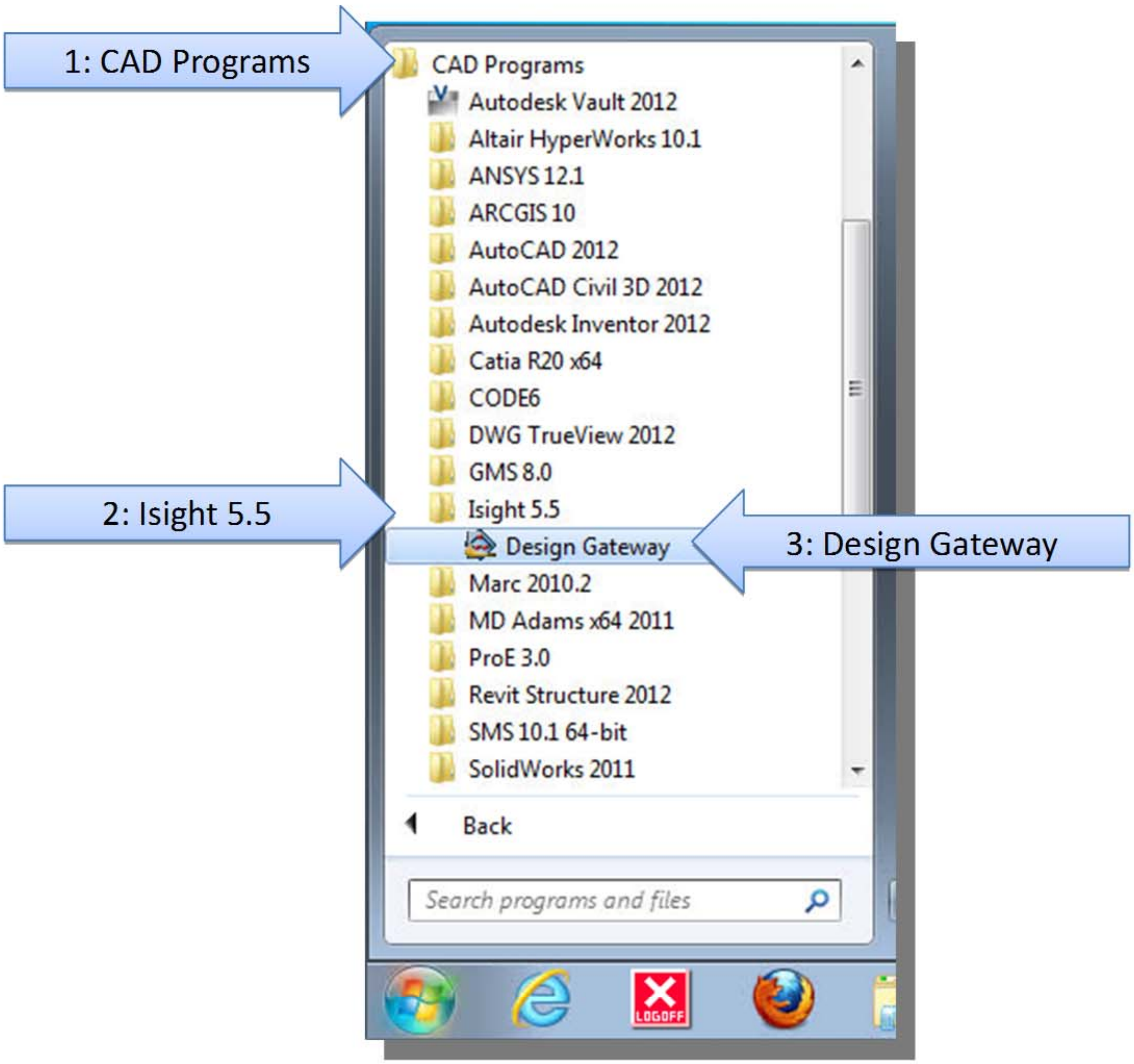
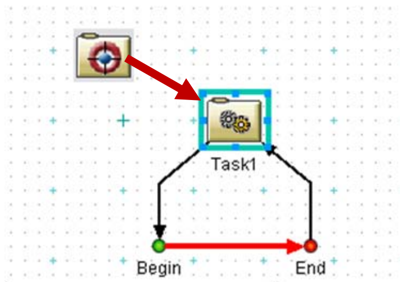
ME 575

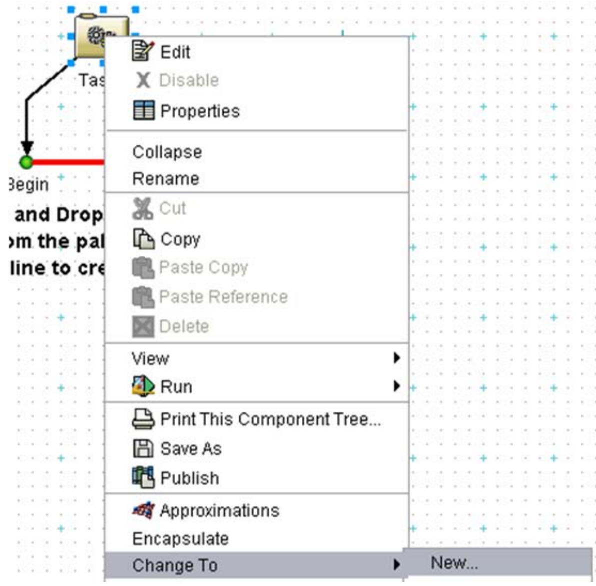
Solving the Twobar Truss Problem Using Isight

1. Log on to an RGS server and launch the Isight 5.5 Design Gateway, as shown below.



2. Create an Optimization Loop by either dragging the **Optimization** component icon  onto the **Task** icon  and confirming the change, or right-click on the **Task** icon then select **Change to…** > **New**, then select the **Optimization** option and click OK.

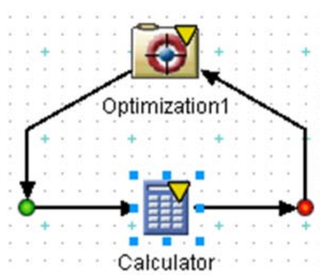




3. From here, there are two basic ways for Isight to compute the constraint and objective functions. The first is with the built-in **Calculator** component  under the *Activities* tab. The second is with a pre-compiled executable that the user creates and connects with Isight using the **Simcode** component , also found under the *Activities* tab.

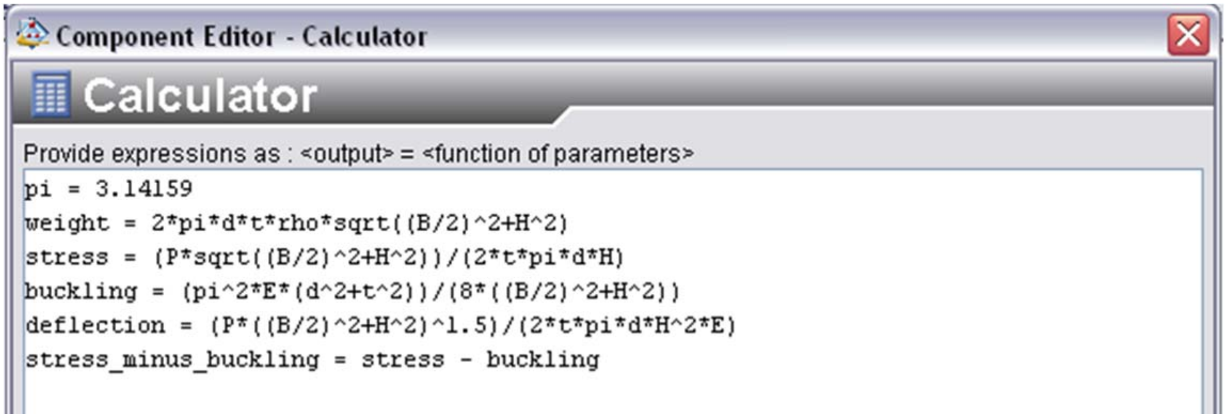
Setting up the Calculator component

1. To use the **Calculator** component, click and drag the icon onto the lower edge of the workflow loop in between the *Begin* and *End* nodes. Your workflow should now look like this:



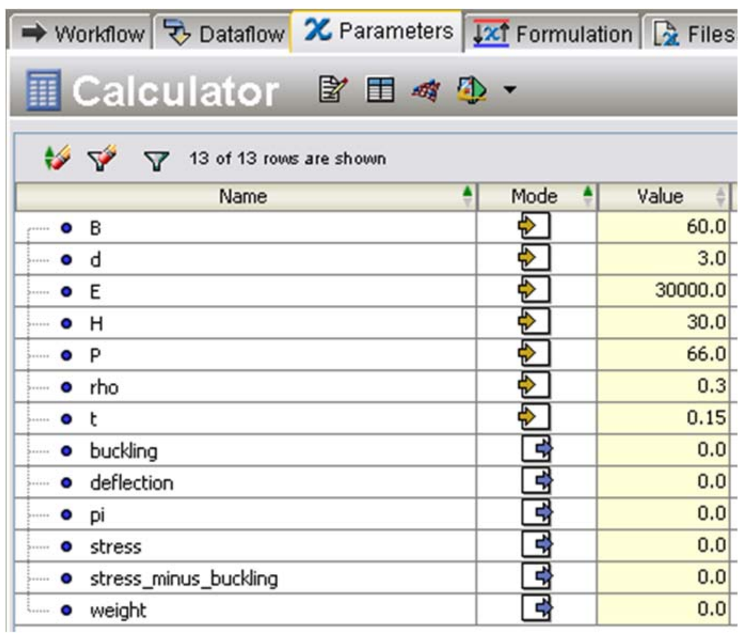
2. Double-click on the **Calculator** icon in the workflow to edit the component.

3. In the editor window, type in the formulas you would like the **Calculator** to compute, using the variable names you desire, following the simple format shown below.



4. Once each formula is free of syntax errors (other than the undefined variables), click OK to close the editor.

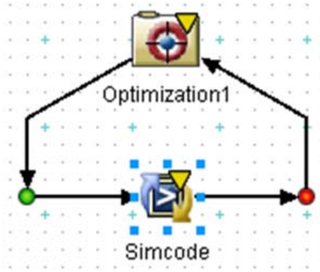
5. Click on the *Parameters* tab near the top of the screen and you will see fields where you can edit the values of the variables you created. Enter the values of the input variables in the **value** field. The input variables are signified by an orange arrow in the *mode* field while the output variables have a blue arrow.



6. The **Calculator** component is now complete. Continue now to the *Setting up the Optimization Component* section below.

Setting up the Simcode component

1. To use the **Simcode** component, click and drag the icon onto the lower edge of the workflow loop in between the *Begin* and *End* nodes. Your workflow should now look like this:

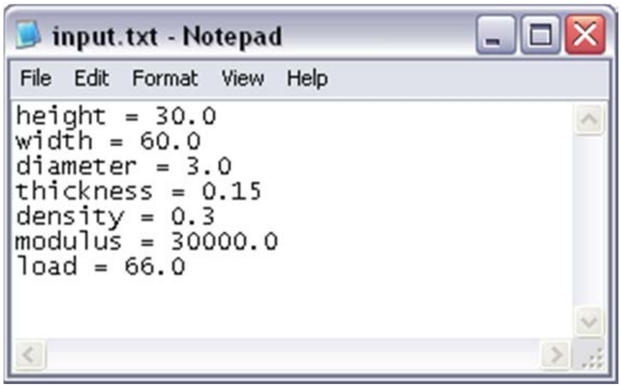


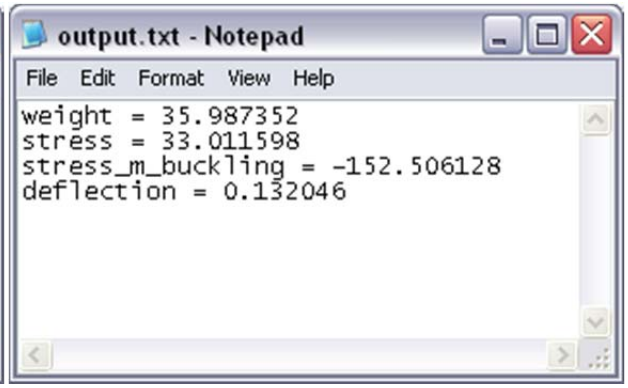
2. Before this component can be utilized, a extra work must be done.
    a. Create an executable that reads from an input file, performs the necessary calculations, then writes to an output file similar to the one below.

```
// TwoBarTruss.cpp : Defines the entry point for the console application.
//
// refer to pg 4-17 of OptdesX Manual for similar example code.
//
#include "stdafx.h"
#include <cmath>
#include <stdio.h>
#include <stdlib.h>

#define PI 3.1415927

int main(int argc, char* argv[])
{
	// input variables
	double height,width,diameter,thickness,density,modulus,load;

	// output function values
	double weight,stress,stress_m_buckling,deflection;

	// other calculated variables
	double length,area,iOverA,buckling;

	// files
	FILE *in, *out;

	// read in variable values
	in = fopen("input.txt","r");
	fscanf(in,"height = %lf\n",&height);
	fscanf(in,"width = %lf\n",&width);
	fscanf(in,"diameter = %lf\n",&diameter);
	fscanf(in,"thickness = %lf\n",&thickness);
	fscanf(in,"density = %lf\n",&density);
	fscanf(in,"modulus = %lf\n",&modulus);
	fscanf(in,"load = %lf\n",&load);
	fclose(in);

	// calculate some intermediate variables
	length = sqrt(width * width / 4. + height * height);
	area = PI * diameter * thickness;
	iOverA = (diameter * diameter + thickness * thickness) / 8.;

	// calculate functions
	weight = 2. * density * area * length;
	stress = load * length / 2. / area / height;
	buckling = (PI * PI * modulus * iOverA) / (length * length);
	stress_m_buckling = stress - buckling;
	deflection = load * pow(length,3) / modulus / area / (height*height);

	// write function values to output file
	out = fopen("output.txt","w");
	fprintf(out,"weight = %lf\n",weight);
	fprintf(out,"stress = %lf\n",stress);
	fprintf(out,"stress_m_buckling = %lf\n",stress_m_buckling);
	fprintf(out,"deflection = %lf\n",deflection);
	fclose(out);

	system("pause");

	return 0;
}
```

    b. Create an input file that contains the input variables similar to the one below. Remember to follow the **name=value** format for each variable.

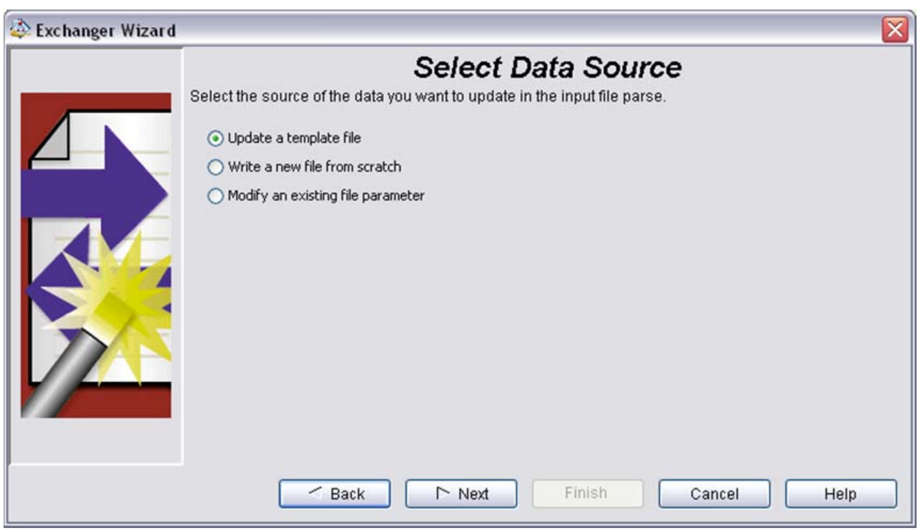
c. Create an output file that contains any calculated function value you might want in the **name=value** format similar to the one shown below.



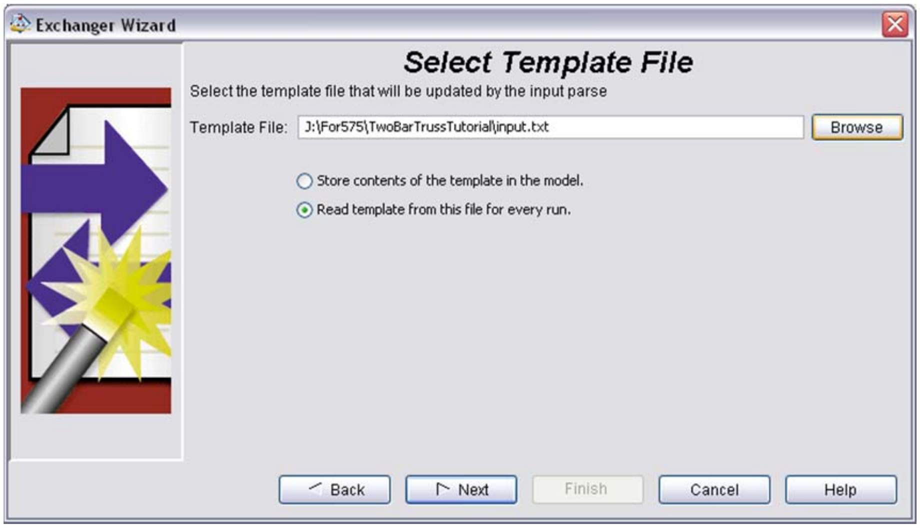


3. Double-click on the **Simcode** icon in the workflow to edit the component.
4. Click on the *Input* tab, then the big **Click here to open a new data source** button.

5. Select *Update a template file*, then click NEXT.

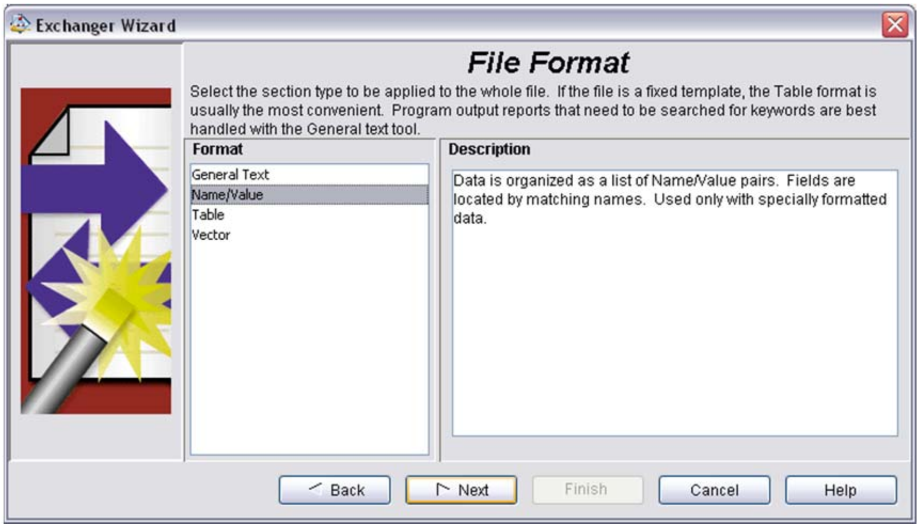


6. Browse to the location of the input file and select the *Read template from this file for every run* radio button, then click NEXT.

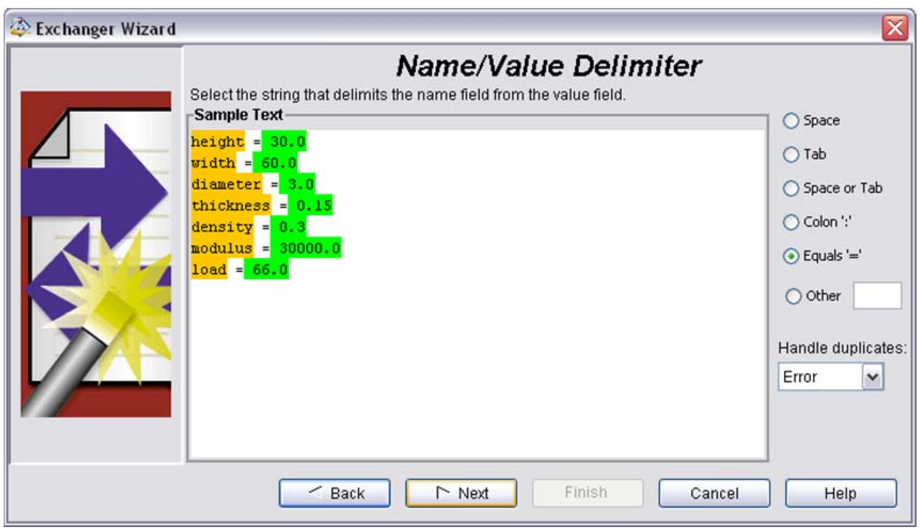


7. The *Local File Name* field should automatically have the name of the input file without the full path. This is correct. Simply click NEXT.

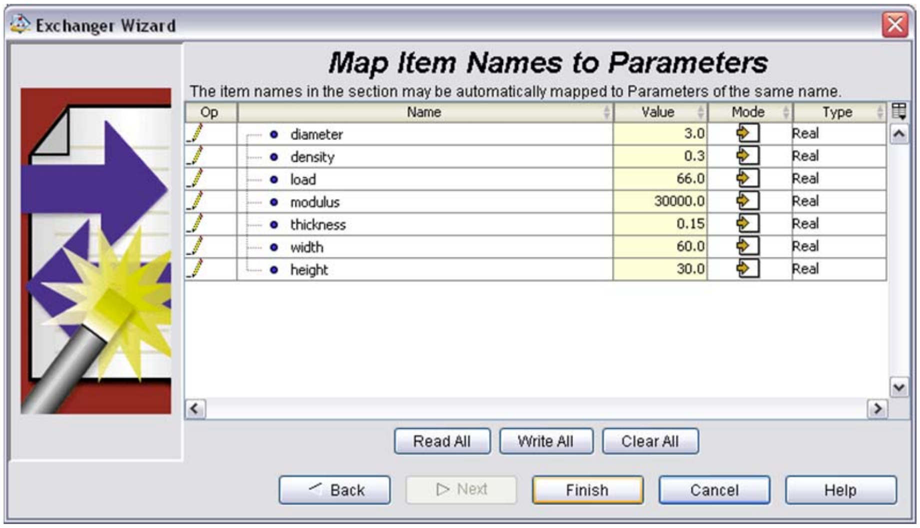
8. Select *Name/Value* in the format window, then click NEXT.



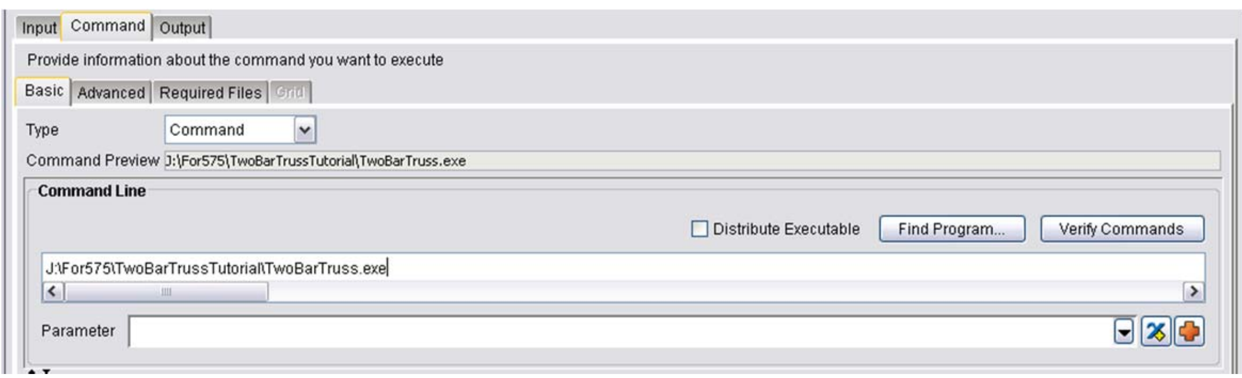
9. You should now see a highlighted form of your input file. Make sure the *Equals* '=' radio button is selected. You should see something similar to what is shown below.



10. The next screen will show you the variables that were created along with their values from the input file. Feel free to modify any nominal value here, then click FINISH to close the wizard.

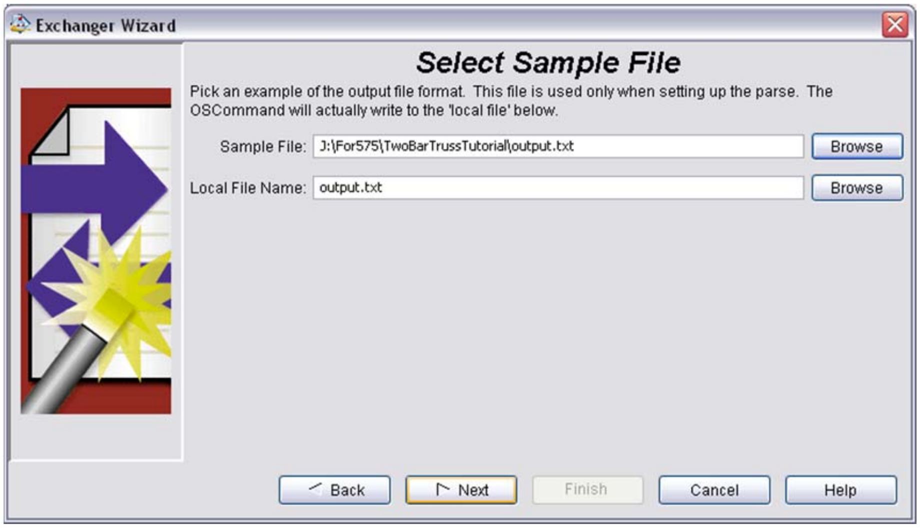


11. Now click on the *Command* tab. Make sure you are in the *Basic* section. In the field provided, type in the full path to the executable you will run, or click on *Find Program...* and browse to it.

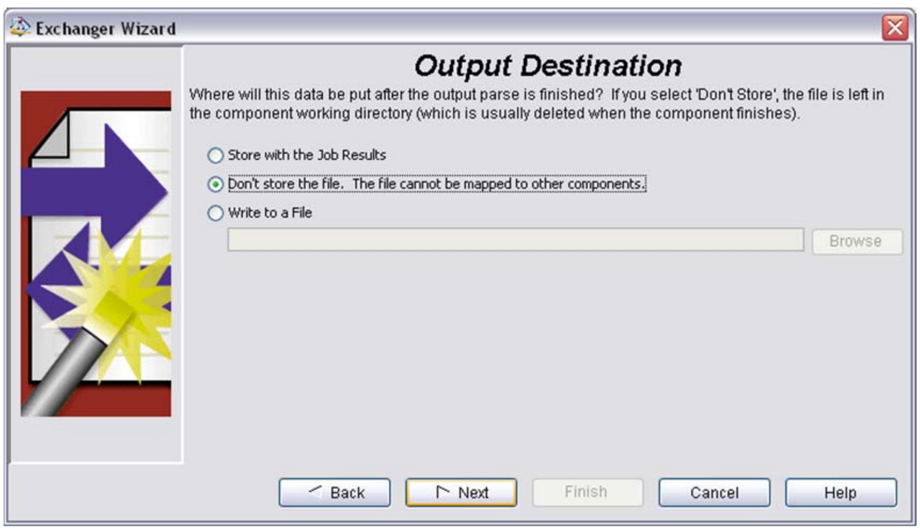


12. Now click on the *Output* tab. A similar process will be followed as the input file. Start by clicking the large **Click here to open a new data source** button.

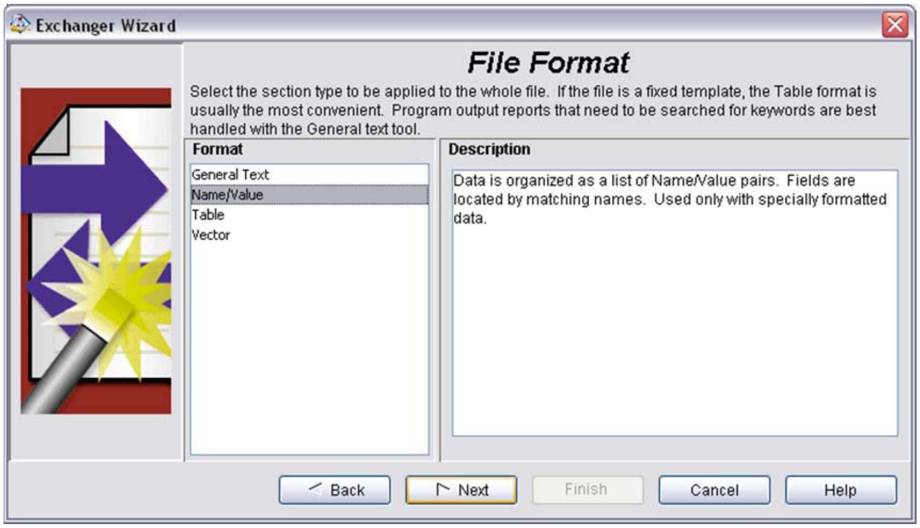
13. Next to the *Sample File* field, click the BROWSE button to select your output file. The *Local File Name* will automatically grab the file name, then click NEXT.



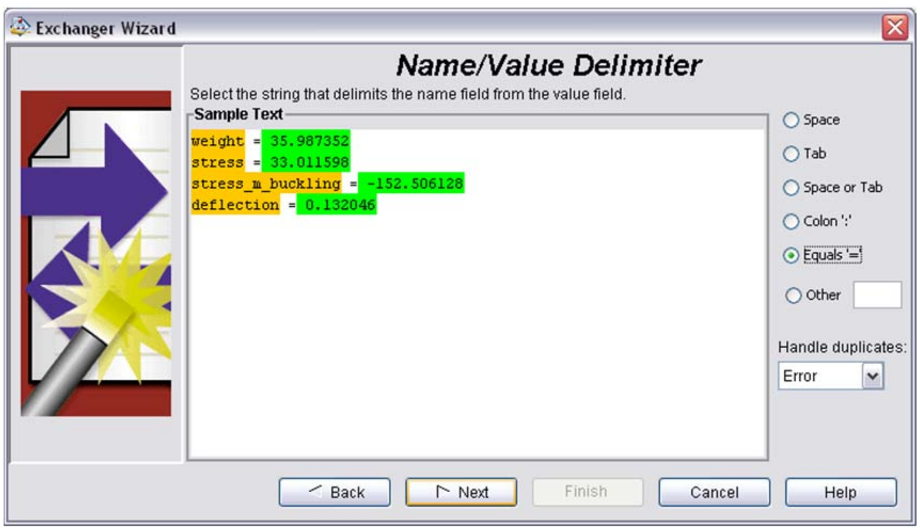
14. Select the radio button *Don't store the file. The File cannot be mapped to other components*, then click NEXT.



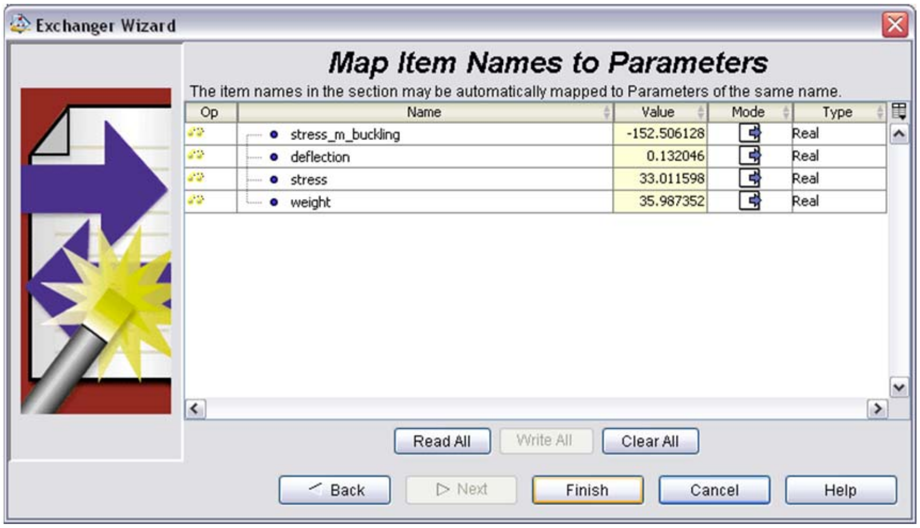
15. Select the *Name/Value* option in the *Format* window and click NEXT.



16. Select the *Equals* '=' radio button, if not already selected and click NEXT.



17. You will see the output variables that have been created along with the values Isight found in the output file. Don't worry about changing these values as they will be modified by the optimization routine. Click FINISH to close the wizard.



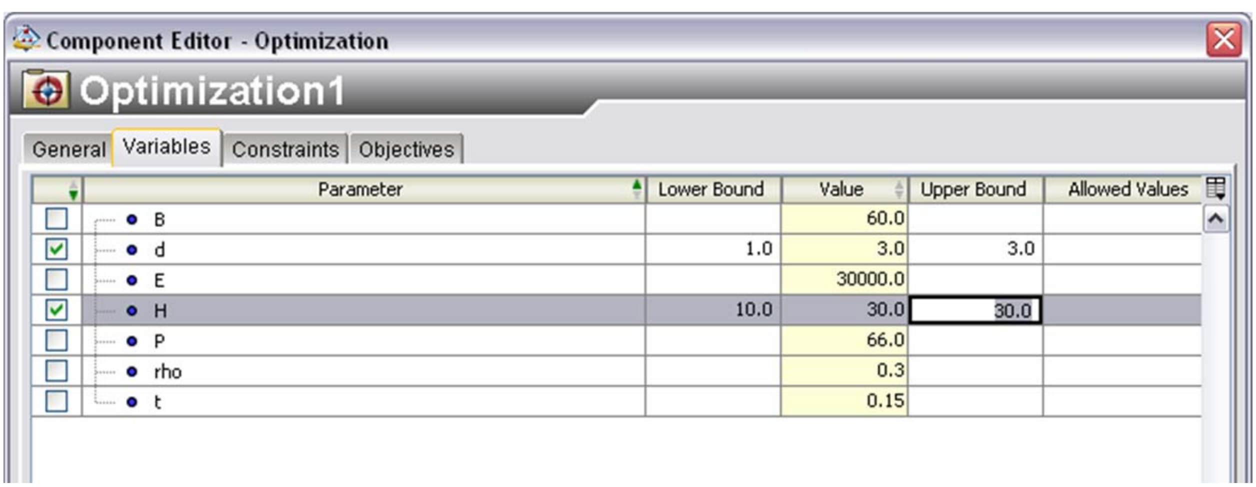
18. The **Simcode** component is now complete. Click OK to close it. Now continue to the *Setting up the Optimization component* section.

Setting up the Optimization component

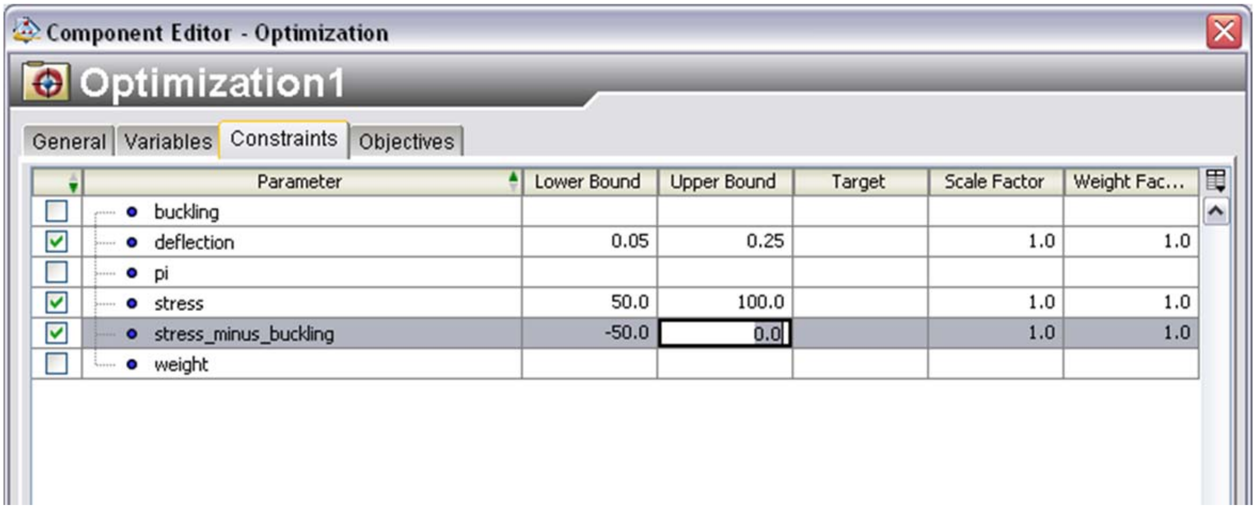
1. For this section, I will use the variables created by the Calculator component, as shown above (the same process is followed for the Simcode component's variables). Click on the *Workflow* tab to return to the original screen and double-click on the **Optimization** component to edit its functionality.

2. In the *General* tab, select **LSGRG** as the *Optimization Technique*. We will use its default settings in this example.

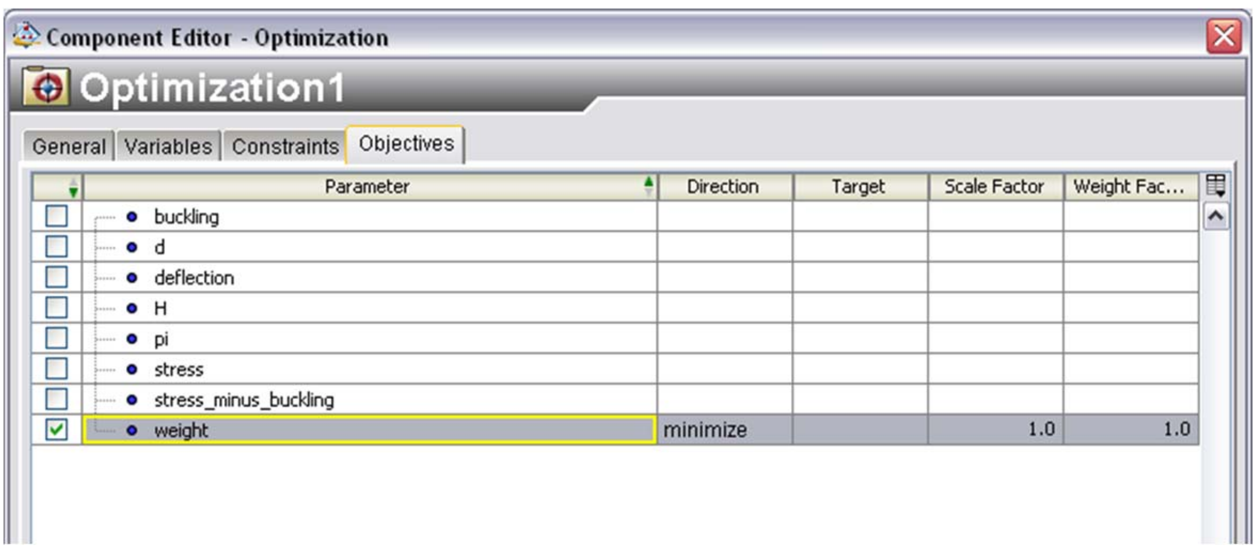
3. In the *Variables* tab, select the check boxes next to diameter (d) and height (H), and modify the lower and upper bounds of each as shown below.



4. In the *Constraints* tab, select the check boxes next to *deflection*, *stress*, and *stress_minus_buckling*, and input each constraint upper and lower bound.

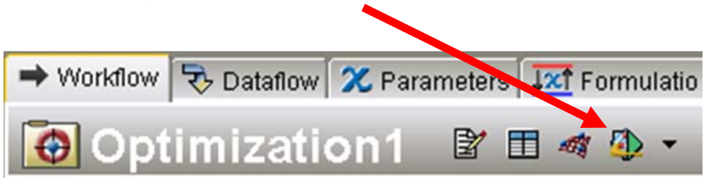


5. In the *Objectives* tab, select the check box next to weight. The default direction for objectives is *minimize*. Click OK and we are now ready to solve.

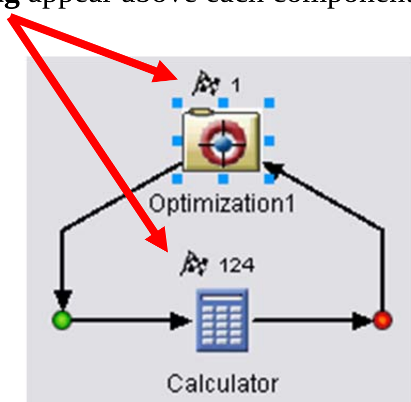


Running the Optimization

1. To execute the workflow, click on the **Run** button.

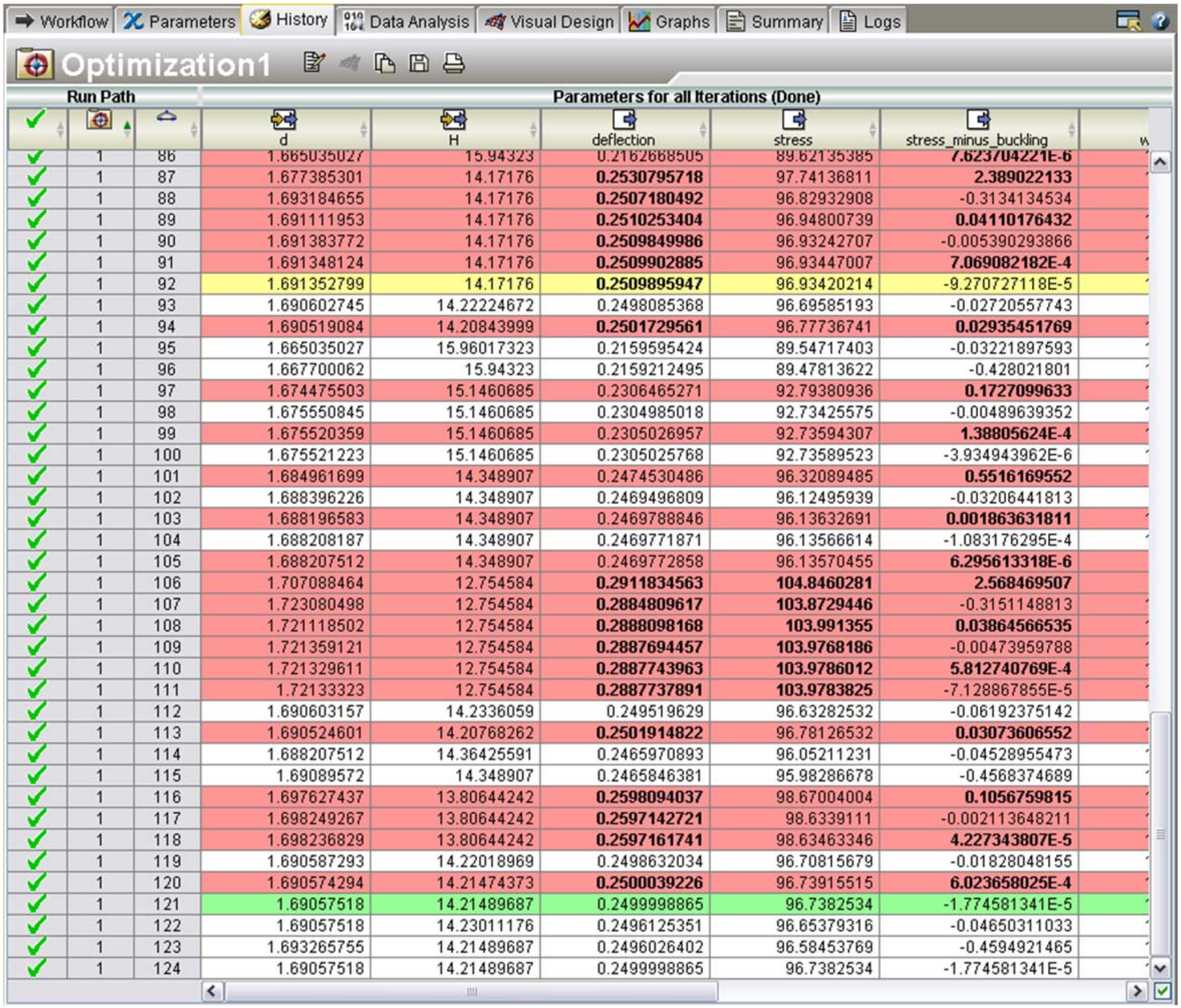


You will then see the *Runtime Gateway* display, showing the workflow and will automatically begin iterating. You know the execution is successful and complete when you see a **checkered-flag** appear above each component in the workflow.



Viewing the results

1. To see what happened in the optimization, click on the *History* tab to see the optimization loop's history of input values and output values. The green row shows the optimum and the red rows show times when constraints were violated.



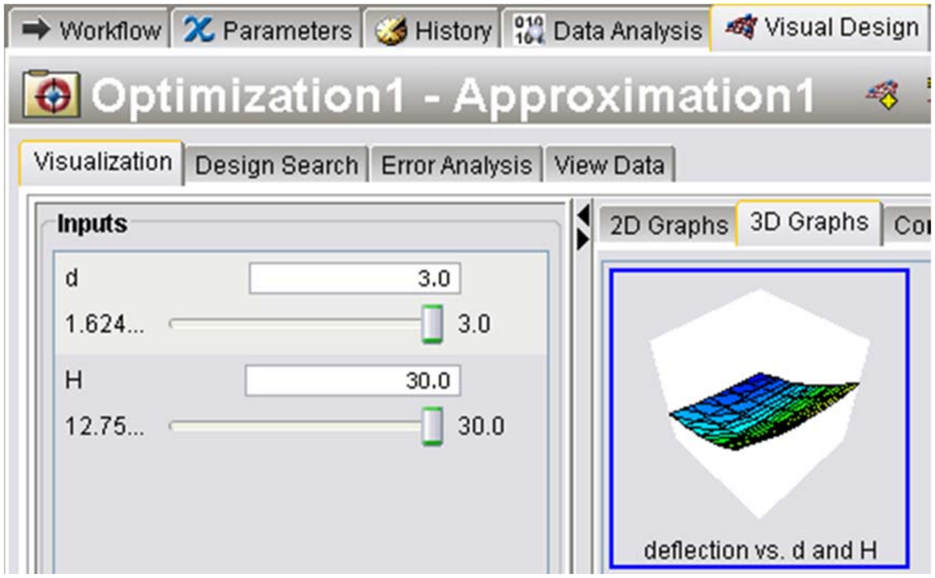
Workflow | Parameters | History | Data Analysis | Visual Design | Graphs | Summary | Logs

Optimization1

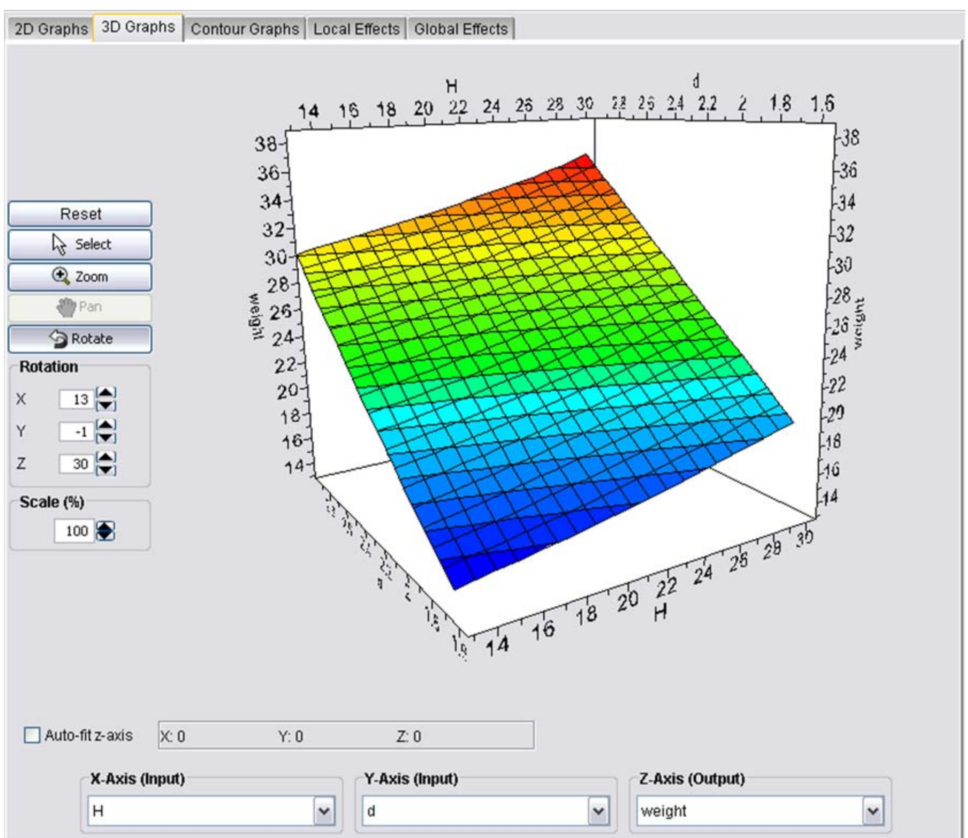
Run Path			Parameters for all Iterations (Done)				
✓			d	H	deflection	stress	stress_minus_buckling
✓	1	86	1.665035027	15.94323	0.2162668505	89.62135385	**7.623704221E-6**
✓	1	87	1.677385301	14.17176	**0.2530795718**	97.74136811	**2.389022133**
✓	1	88	1.693184655	14.17176	**0.2507180492**	96.82932908	-0.3134134534
✓	1	89	1.691111953	14.17176	**0.2510253404**	96.94800739	**0.04110176432**
✓	1	90	1.691383772	14.17176	**0.2509849986**	96.93242707	-0.005390293866
✓	1	91	1.691348124	14.17176	**0.2509902885**	96.93447007	**7.069082182E-4**
✓	1	92	1.691352799	14.17176	**0.2509895947**	96.93420214	-9.270727118E-5
✓	1	93	1.690602745	14.22224672	0.2498085368	96.69585193	-0.02720557743
✓	1	94	1.690519084	14.20843999	**0.2501729561**	96.77736741	**0.02935451769**
✓	1	95	1.665035027	15.96017323	0.2159595424	89.54717403	-0.03221897593
✓	1	96	1.667700062	15.94323	0.2159212495	89.47813622	-0.428021801
✓	1	97	1.674475503	15.1460685	0.2306465271	92.79380936	**0.1727099633**
✓	1	98	1.675550845	15.1460685	0.2304985018	92.73425575	-0.00489639352
✓	1	99	1.675520359	15.1460685	0.2305026957	92.73594307	**1.38805624E-4**
✓	1	100	1.675521223	15.1460685	0.2305025768	92.73589523	-3.934943962E-6
✓	1	101	1.684961699	14.348907	0.2474530486	96.32089485	**0.5516169552**
✓	1	102	1.688396226	14.348907	0.2469496809	96.12495939	-0.03206441813
✓	1	103	1.688196583	14.348907	0.2469788846	96.13632691	**0.001863631811**
✓	1	104	1.688208187	14.348907	0.2469771871	96.13566614	-1.083176295E-4
✓	1	105	1.688207512	14.348907	0.2469772858	96.13570455	**6.295613318E-6**
✓	1	106	1.707088464	12.754584	**0.2911834563**	**104.8460281**	**2.568469507**
✓	1	107	1.723080498	12.754584	**0.2884809617**	**103.8729446**	-0.3151148813
✓	1	108	1.721118502	12.754584	**0.2888098168**	**103.991355**	**0.03864566535**
✓	1	109	1.721359121	12.754584	**0.2887694457**	**103.9768186**	-0.00473959788
✓	1	110	1.721329611	12.754584	**0.2887743963**	**103.9786012**	**5.812740769E-4**
✓	1	111	1.72133323	12.754584	**0.2887737891**	**103.9783825**	-7.128867855E-5
✓	1	112	1.690603157	14.2336059	0.249519629	96.63282532	-0.06192375142
✓	1	113	1.690524601	14.20768262	**0.2501914822**	96.78126532	**0.03073606552**
✓	1	114	1.688207512	14.36425591	0.2465970893	96.05211231	-0.04528955473
✓	1	115	1.69089572	14.348907	0.2465846381	95.98286678	-0.4568374689
✓	1	116	1.697627437	13.80644242	**0.2598094037**	98.67004004	**0.1056759815**
✓	1	117	1.698249267	13.80644242	**0.2597142721**	98.6339111	-0.002113648211
✓	1	118	1.698236829	13.80644242	**0.2597161741**	98.63463346	**4.227343807E-5**
✓	1	119	1.690587293	14.22018969	0.2498632034	96.70815679	-0.01828048155
✓	1	120	1.690574294	14.21474373	**0.2500039226**	96.73915515	**6.023658025E-4**
✓	1	121	1.69057518	14.21489687	0.2499998865	96.7382534	-1.774581341E-5
✓	1	122	1.69057518	14.23011176	0.2496125351	96.65379316	-0.04650311033
✓	1	123	1.693265755	14.21489687	0.2496026402	96.58453769	-0.4594921465
✓	1	124	1.69057518	14.21489687	0.2499998865	96.7382534	-1.774581341E-5

2. To see plots of the design space, first click on the *Visual Design* tab then on the “Create Approximation” button.

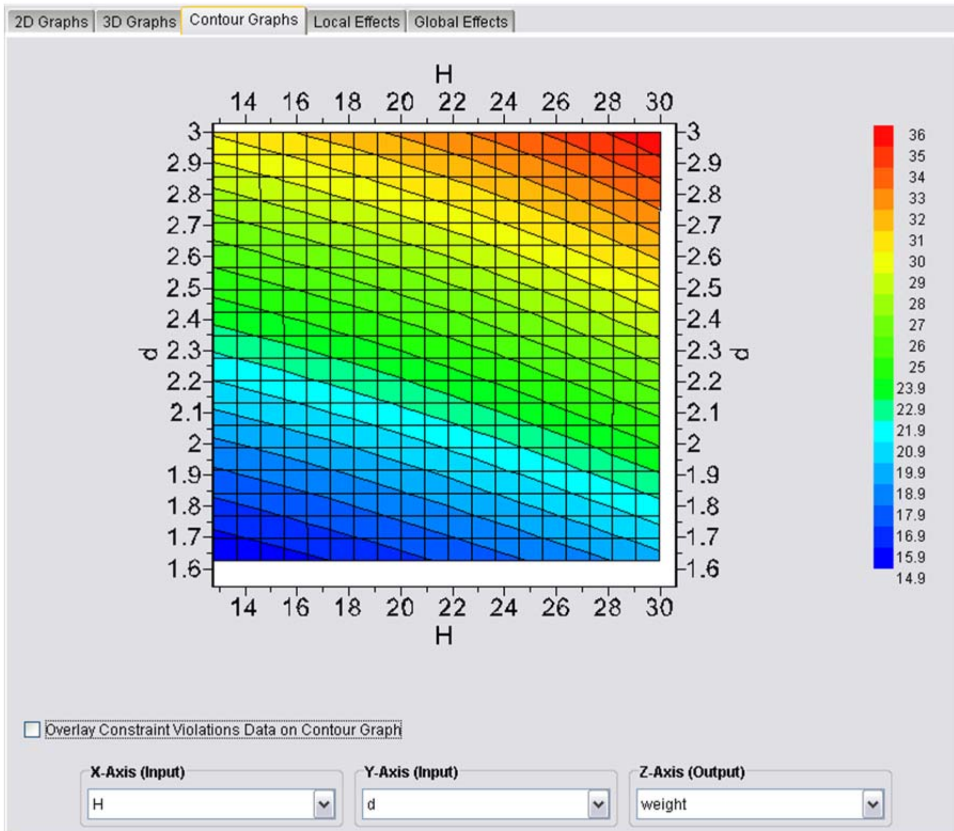
3. You can then view the data in a myriad of ways, including 3D surface plots, 2D contour plots, correlation graphs, etc.
    a. 3D surface plots are found under the *Visualization* tab, then the *3D Graphs* tab. Double click on any image to open the plot.



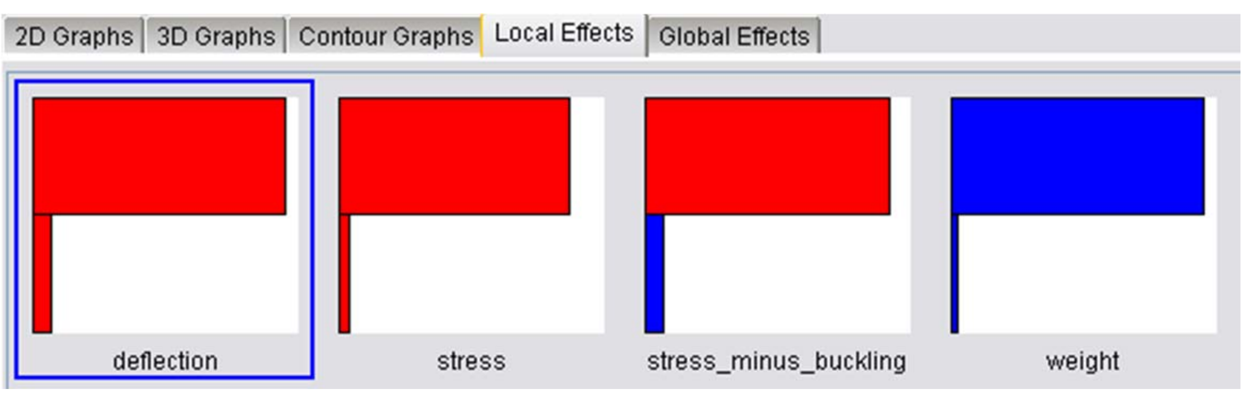
b. Plot controls are on the left, allowing you to zoom, pan, and rotate the plot. You may change the plot input variables (X-axis and Y-axis) and output variable (Z-axis) below the displayed plot. Double-clicking on the plot will return to the previous selection screen.



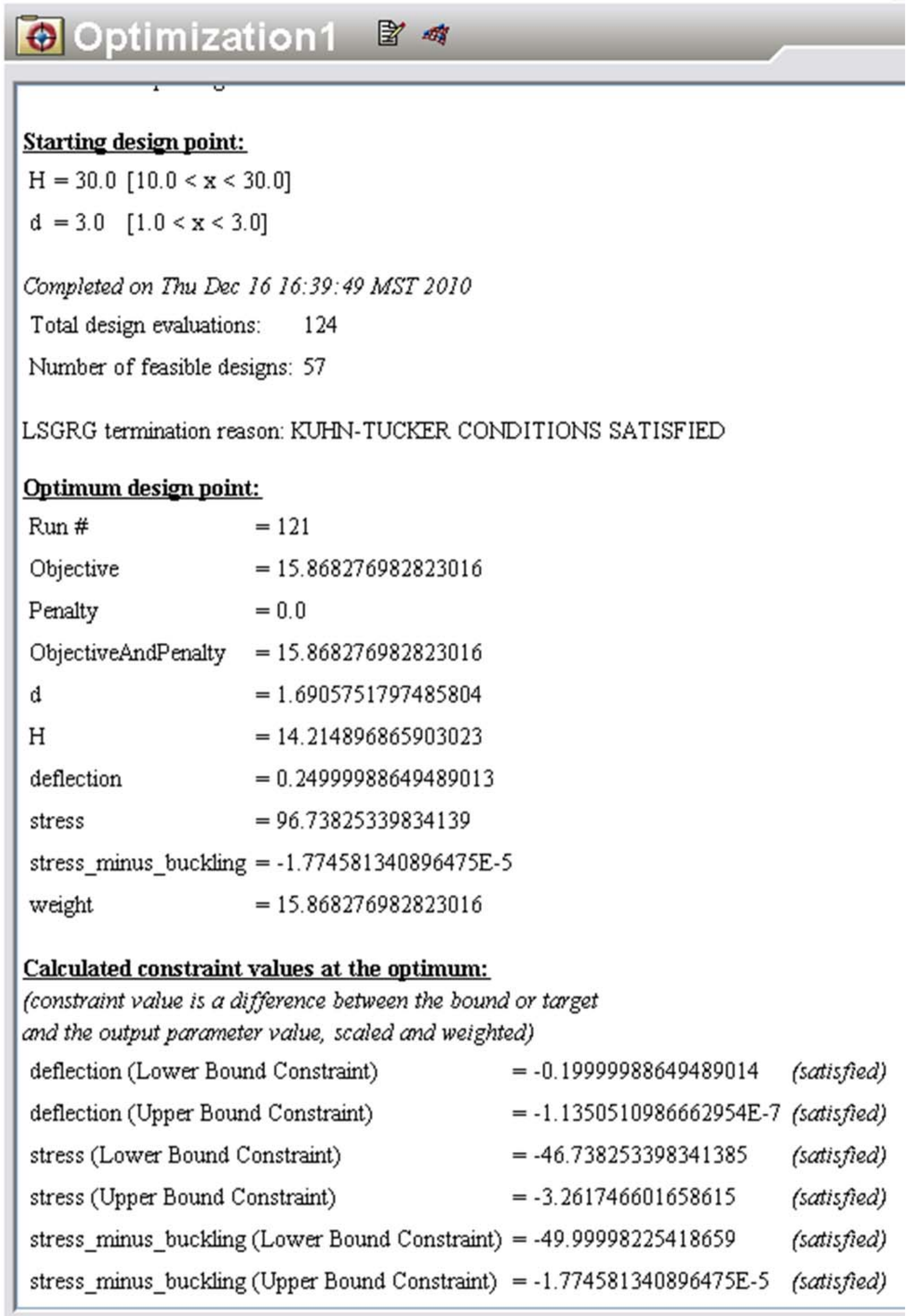
c. 2D contour plots are found under the *Contour Graphs* tab and have similar functionality as the 3D plots.



d. Clicking on the *Local Effects* or *Global Effects* tabs, then double-clicking any output variable allows the user to discover, based on the optimization, how much each variable affects that particular output variable, both constraints and objectives.



e. Click on the *Summary* tab to view a textual optimization summary, including termination reason (here the Kuhn-Tucker conditions were satisfied) along with the optimum and constraint values at the optimum.



Starting design point:

H = 30.0 [10.0 < x < 30.0]

d = 3.0 [1.0 < x < 3.0]

Completed on Thu Dec 16 16:39:49 MST 2010

Total design evaluations: 124

Number of feasible designs: 57

LSGRG termination reason: KUHN-TUCKER CONDITIONS SATISFIED

Optimum design point:

Run #	= 121
Objective	= 15.868276982823016
Penalty	= 0.0
ObjectiveAndPenalty	= 15.868276982823016
d	= 1.6905751797485804
H	= 14.214896865903023
deflection	= 0.24999988649489013
stress	= 96.73825339834139
stress_minus_buckling	= -1.774581340896475E-5
weight	= 15.868276982823016

Calculated constraint values at the optimum:

(constraint value is a difference between the bound or target and the output parameter value, scaled and weighted)

deflection (Lower Bound Constraint)	= -0.19999988649489014	*(satisfied)*
deflection (Upper Bound Constraint)	= -1.1350510986662954E-7	*(satisfied)*
stress (Lower Bound Constraint)	= -46.738253398341385	*(satisfied)*
stress (Upper Bound Constraint)	= -3.261746601658615	*(satisfied)*
stress_minus_buckling (Lower Bound Constraint)	= -49.99998225418659	*(satisfied)*
stress_minus_buckling (Upper Bound Constraint)	= -1.774581340896475E-5	*(satisfied)*