

Athena Visual Studio Nonlinear Optimization Tutorial

- ❖ Start Athena Visual Studio
- ❖ From the **File** menu select **New**.
- ❖ You are in the **Process Modeling** tab.
- ❖ Select the **Nonlinear Optimization** tab.
- ❖ Click **Optimization with User Defined Explicit Models**.
- ❖ Select **A Blank Document** and click **OK**.
- ❖ Enter your model data, objective function and constraints, and the Athena solver data and options as described in this tutorial.

When you are done:

- ❖ From the **File** menu click **Save**.
- ❖ Navigate to the folder where that you wish to save and enter a proper filename for your model.
- ❖ From the **Build** menu click **Compile**.
- ❖ From the **Build** menu click **Build EXE**.
- ❖ From the **Build** menu click **Execute**.

The basic elements of an optimization problem are listed below:

1. The objective [or merit] function
2. The equality constraints
3. The inequality constraints
4. The lower and upper bounds on the independent variables
5. A set of independent or decision variables
6. A set of known parameters pertinent to the process under investigation

The conventional mathematical formulations for various types

of optimization problems including both constrained and unconstrained are given in the tables below. In these tables we differentiate between explicit and implicit models. Implicit models frequently contains mixed systems of differential and algebraic equations, that need to be solved efficiently during the optimization process. Differential equations can be both ordinary and partial, thus describing a wide range of lumped as well distributed parameter systems, such as continuous stirred tank reactors, fixed bed and plug flow reactors both dynamic and steady-state.

1. Optimization of Explicit Models:

Constrained Optimization		Unconstrained Optimization	
Objective	$\min_{\mathbf{x}} f(\mathbf{x}; \theta)$	$\min_{\mathbf{x}} f(\mathbf{x}; \theta)$	$\min_{\mathbf{x}} f(\mathbf{x}; \theta)$
Equality Constraints	$\mathbf{h}(\mathbf{x}; \theta) = \mathbf{0}$	$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$	$-\infty \leq \mathbf{x} \leq +\infty$
Inequality Constraints	$\mathbf{c}(\mathbf{x}; \theta) \geq \mathbf{0}$		
Variable Bounds	$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$		

2. Optimization of Implicit Differential/Algebraic Models:

Objective	$\min_{\mathbf{x}} f(\mathbf{x}; \theta)$	State Equations	$\mathbf{E}(t, \mathbf{u}; \theta) \frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{x}, \mathbf{u}; \theta)$
Equality Constraints	$\mathbf{h}(\mathbf{x}; \theta) = \mathbf{0}$	Initial Conditions	$t = t_0 \quad \mathbf{u} = \mathbf{u}_0(\theta)$
Inequality Constraints	$\mathbf{c}(\mathbf{x}; \theta) \geq \mathbf{0}$	Connectivity Constraints	$\mathbf{q}(t, \mathbf{x}, \mathbf{u}; \theta) = \mathbf{0}$
Variable Bounds	$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$		

where

\mathbf{x}	Multi-dimensional vector of unknown variables also known as <i>independent variables</i> or <i>decision variables</i>
\mathbf{u}	Multi-dimensional vector of state variables
θ	Multi-dimensional vector of known parameters
$f(\mathbf{x}; \theta)$	The [scalar] objective function ➤ <i>Profit or cost financial objective</i> ➤ <i>Process objective</i>
$\mathbf{h}(\mathbf{x}; \theta) = \mathbf{0}$	Multi-dimensional vector of equality constraints ➤ <i>Material and Energy balance equations</i> ➤ <i>Design specifications</i>
$\mathbf{c}(\mathbf{x}; \theta) \geq \mathbf{0}$	Multi-dimensional vector of inequality constraints ➤ <i>Design and Equipment constraints</i> ➤ <i>Process constraints</i>
$\mathbf{x}^L, \mathbf{x}^U$	Lower and upper bounds on the independent variables

A point \mathbf{x} that satisfies the equality and inequality constraints is called a **feasible point**. The set of all points that are feasible is called the **feasible region**.

Objective Function: The objective function usually is a profit or cost function. Frequently the objective function represents a process variable, such as, production rate.

Equality Constraints: The equality constraints typically consist of the material and energy balance, and miscellaneous design and operating specifications.

Inequality Constraints: The inequality constraints represent design and operation limitations as well as operating regions.

Lower and Upper Bounds: Lower and upper bounds on the independent variables represent feasible operating regimes.

Tutorial: Alkylation Process Optimization

The following nonlinear model including the objective function and process constraints has been postulated for the optimization of the alkylation process:

maximize the process profit

$$\Phi(x) = c_1 x_4 x_7 - c_2 x_1 - c_3 x_2 - c_4 x_3 - c_5 x_5$$

subject to process constraints

$$x_4 = x_1 (1.12 + 0.12167 x_8 - 0.0067 x_8^2)$$

$$x_5 = 1.22 x_4 - x_1$$

$$x_6 (x_4 x_9 + 1000 x_3) = 98000 x_3$$

$$x_7 = 86.35 + 1.098 x_8 - 0.038 x_8^2 + 0.325 (x_6 - 89)$$

$$x_8 x_1 = x_2 + x_5$$

$$x_9 = 35.82 - 0.222 x_{10}$$

$$x_{10} = 3 x_7 - 133$$

Alkylate Yield

Isobutane Makeup

Acid Strength

Motor Octane

Isobutane to Olefin Ratio

Acid Dilution Strength

F-4 Performance

The profit function is calculated based on cost data given in the table below along with the definitions of the Alkylation process variables.

Alkylate Product Value	$C_1 = 0.063$	x_1	Olefin Feed	x_6	Acid Strength
Olefin Feed Cost	$C_2 = 5.04$	x_2	Isobutane Recycle	x_7	Motor Octane Number
Isobutane Feed Cost	$C_3 = 3.36$	x_3	Acid Addition Rate	x_8	External Isobutane to Olefin Ratio
Isobutane Recycle Cost	$C_4 = 0.035$	x_4	Alkylate Yield	x_9	Acid Dilution Factor
Acid Addition Cost	$C_5 = 10.0$	x_5	Isobutane Makeup	x_{10}	F-4 Performance Number of Alkylate

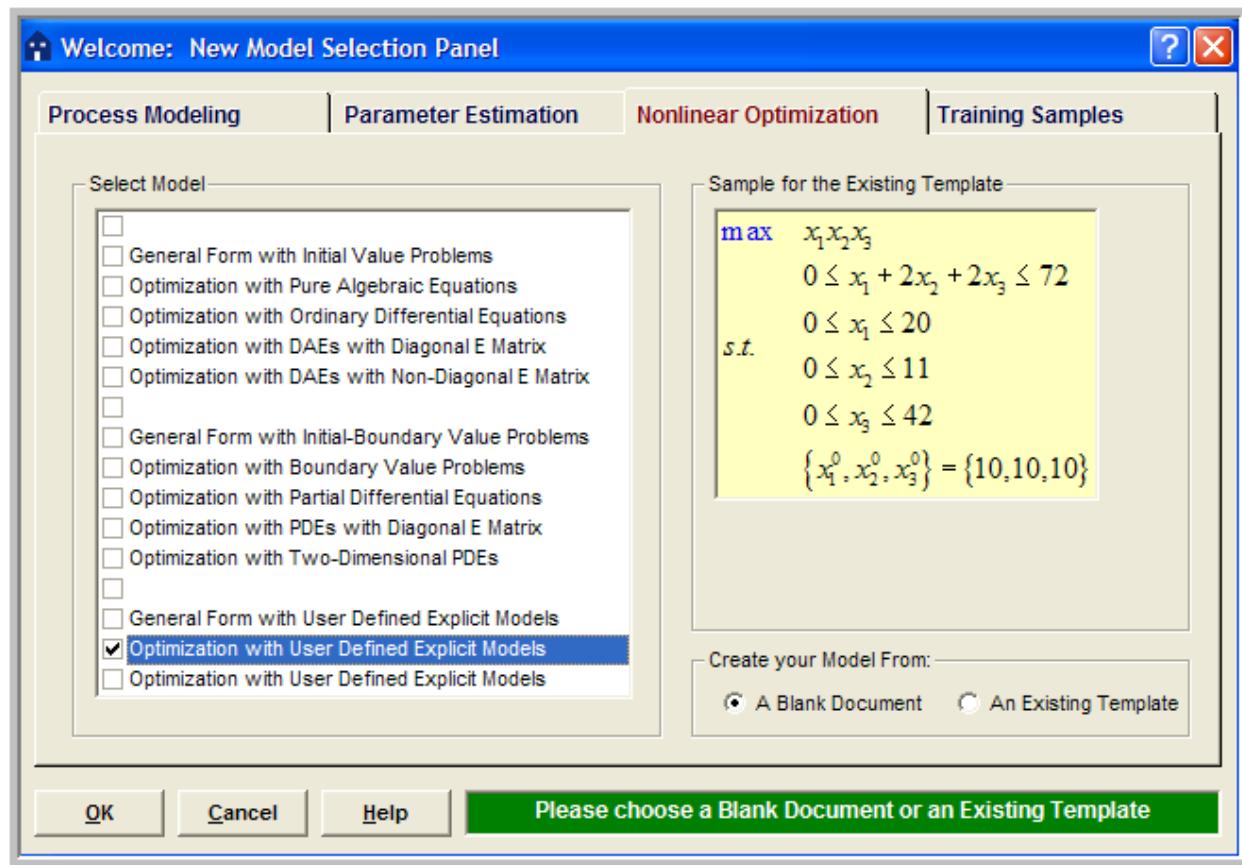
This example tutorial is already pre-coded in Athena Visual Studio. If you do not wish to type the code on your own you may access it by doing the following:

- ❖ Open **Athena Visual Studio**
- ❖ From the **File** menu click **New**
- ❖ Select the **Training Samples** tab
- ❖ Select the **Nonlinear Constrained Optimization** sample
- ❖ Click **OK**

Implementation in Athena Visual Studio

The following step by step process describes the model implementation in Athena Visual Studio

- ❖ Open Athena Visual Studio.
- ❖ From the **File** menu, choose **New**.
- ❖ The *Welcome: New Model Selection Panel* window appears.



- ❖ Select the *Nonlinear Optimization* tab
- ❖ Select the *Optimization with User Defined Explicit Models* option.
- ❖ Choose *A Blank Document* and click **OK**.

Type your source code in the new window. The source code contains standard modeling sections (see description in the next sections below); it may also contain calls to the available math and engineering procedures as well as user-defined procedures.

Writing the Source Code

You must enter a minimum of one section in order to create the nonlinear optimization model with user defined explicit models. This section is labeled **@Optimization Model** and is used to insert the objective function and the constraints. A data section not labeled by Athena Visual Studio may also be used to enter all the data pertinent to the model. The data section may also contain declaration statements for all model variables, parameters and constants. This section, if used, must be the first one in the model. The declaration of the model variables, parameters and constants must be done in accordance the Athena Visual Studio syntax rules shown below:

Declaration of Variables

Global Variables: To declare global variables in the Athena Visual Studio environment you must use the **Global** keyword as the examples below illustrate:

```
Global x, y, z, krate As Real  
Global Skount, Ncc As Integer  
Global myName As Character  
Global myDecision As Logical
```

In the above statements the variables *x, y, z, krate* will be treated as double precision and will be accessible by all modeling sections. Similarly the variables *Skount, Ncc* will be treated as integer and be accessible by all modeling sections. Character variables are assigned as Character*132 from the Athena Visual Studio parser. Single precision variables cannot be declared **Global**.

Vectors and matrices can be declared in a similar manner. The array size, type and number of dimensions are declared with the **Global** statement. The elements of the array can be referenced by an integer index number, which runs from one (or zero) to the maximum number declared in the **Global** statement:

```
Global y(10), c(0:5), a(4,50), b(2,4,6) As Real  
Global istate(5) As Integer
```

Local Variables: To declare local variables in the Athena Visual Studio environment you must use the **Dim** keyword as the examples below illustrate:

```
Dim Temp, Pres As Real  
Dim TotalFlow As Single  
Dim i As Integer
```

In the above statements the variables *Temp, Pres* will be treated as double precision, where as the variable *TotalFlow* will be treated as single precision; these variables will be accessible only at the section where they have been declared. Similarly the variable *i* will be treated as integer and will be accessible only by the corresponding modeling section where it has been declared.

Vectors and matrices can be declared in a similar manner. The array size, type and number of dimensions are declared with the **Dim** statement. The elements of the array are referenced by an integer index, which runs from one(or zero) to the number declared in the **Dim** statement:

```
Dim c(10), p(4,50) As Real  
Dim streamEnthalpy(10) As Single  
Dim irow(5) As Integer
```

Parameter Statement: Use the **Parameter** keyword to define named constants as the examples below illustrate:

```
Parameter y=2.0, z=4.0 As Real  
Parameter Skount=1, Ncc=4 As Integer
```

In the above statements the variables *y*, *z* will be treated as double precision and their numerical values will be accessible by all parts of the modeling code. Similarly the variables *Skount*, *Ncc* will be treated as integer and their numerical values will be accessible through out all the modeling sections. The **Parameter** keyword is only allowed in the data section of the Athena Visual Studio modeling code. If it is used in the other modeling sections it will be ignored. You may view the generated Fortran code to see how the parser interprets the **Parameter** keyword.

Important Note: Always remember to declare all of your variables. Athena treats **Real** variables as double precision, **Integer** variables as 4-byte integers, **Character** variables as Character*132 and **Logical** variables as **.True.** or **.False.** Single precision variables are only allowed if are declared as local with the **Dim** keyword.

Fortran 95 Declaration Statements: You can insert Fortran 95 declaration statements by prefixing them with the double dollar sign. Below please see a list of Fortran 95 declaration statements that you can insert in your Athena code. Consult your Fortran 95 manual for the syntax rules of variable and constant declarations:

```
$$Integer, Parameter::: dp=Kind(1.0D0)  
$$Integer, Parameter::: sp=Kind(1.0)  
$$Real(Kind=dp):: v1,v2  
$$Real(Kind=sp), Dimension(3):: a1,a2  
$$Integer:: I1, I2  
$$Character(Len=3):: s2,s3  
$$Character(Len=10), Dimension(2):: s1  
$$Logical:: Done  
$$Real(Kind=dp), Dimension(:), Allocatable:: w
```

We are now going to describe in detail the various steps involved in writing an explicit model for nonlinear optimization in the Athena Visual Studio environment. The modeling code is NOT case sensitive.

Data Section

In the data section the user simply enters the problem data and various constants. The data section also contains the declarations of problem variables, parameters and constants. For our example the user enters the cost data required for the calculation of the objective function. The Athena interpreter treats any line that begins with an exclamation mark ! as a comment. It is mandatory and strongly recommended that the users declare all the problem parameters and constants. All variables in Athena are either real double or single precision or integer long. Character and Logical variables are also allowed. The following source code may be entered for this example (If you use the Windows Copy and Paste commands to enter this code into your Athena Visual Studio project, please beware that invisible format symbols may also be copied and cause the compilation of your code to fail):

```
! Declarations and Model Constants
!-----
Global C1,C2,C3,C4,C5 As Real

C1=0.063  ! alkylate product value/octane-barrel
C2=5.04   ! olefin feed cost/barrel
C3=0.035  ! isobutane recycle costs/barrel
C4=10.0   ! acid addition cost/1000 lbs
C5=3.36   ! isobutane make-up cost/barrel
```

Optimization Model

In the Optimization Model section the user must enter the objective function and the constraints(if any). The vector **F()** is reserved to enter these functions. For example **F(1)** represents the objective function, **F(2)** represents the equality constraint that describes the alkylate yield and so on. To enter the objective and constraints for your model:

- ❖ From the **Model** menu choose **Optimization Model** (or **Hit F11**)
- ❖ Enter the source code as shown below for our example.

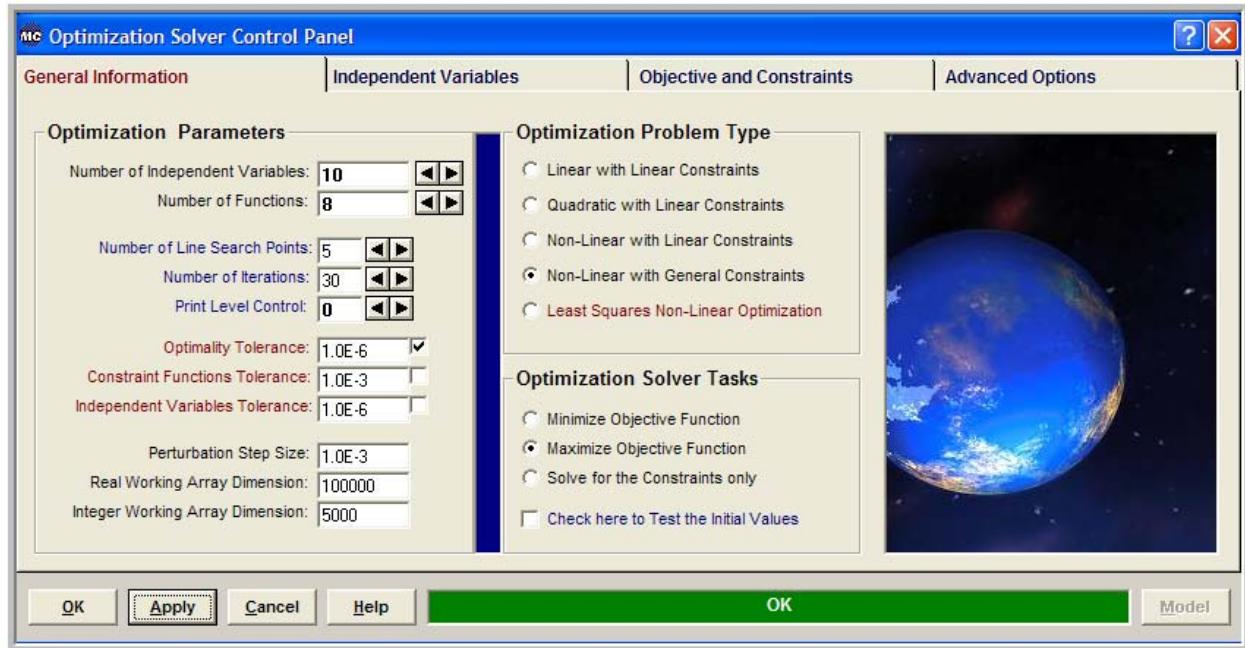
```
@Optimization Model
F(1)=C1*X(4)*X(7)-C2*X(1)-C3*X(2)-C4*X(3)-C5*X(5)

F(2)=X(4)-X(1)*(1.12+0.12167*X(8)-0.00667*X(8)^2)
F(3)=X(5)+X(1)-1.22*X(4)
F(4)=X(6)*(X(4)*X(9)+1000.0*X(3))-98000.0*X(3)
F(5)=X(7)-1.098*X(8)+0.0038*X(8)^2-0.325*(X(6)-89.0)
F(6)=X(8)*X(1)-X(2)-X(5)
F(7)=X(9)+0.222*X(10)
F(8)=3.0*X(7)-X(10)
```

The Optimization Solver

It is now time to access the Athena Visual Studio solver for Nonlinear Optimization in order to enter information about the independent variables, the objective function and constraints and various other parameters that control the optimization algorithm. To do that:

- ❖ From the **Model** menu choose **Load Solver** (or **Hit F12**)
- ❖ Enter the solver parameters as shown below for our example



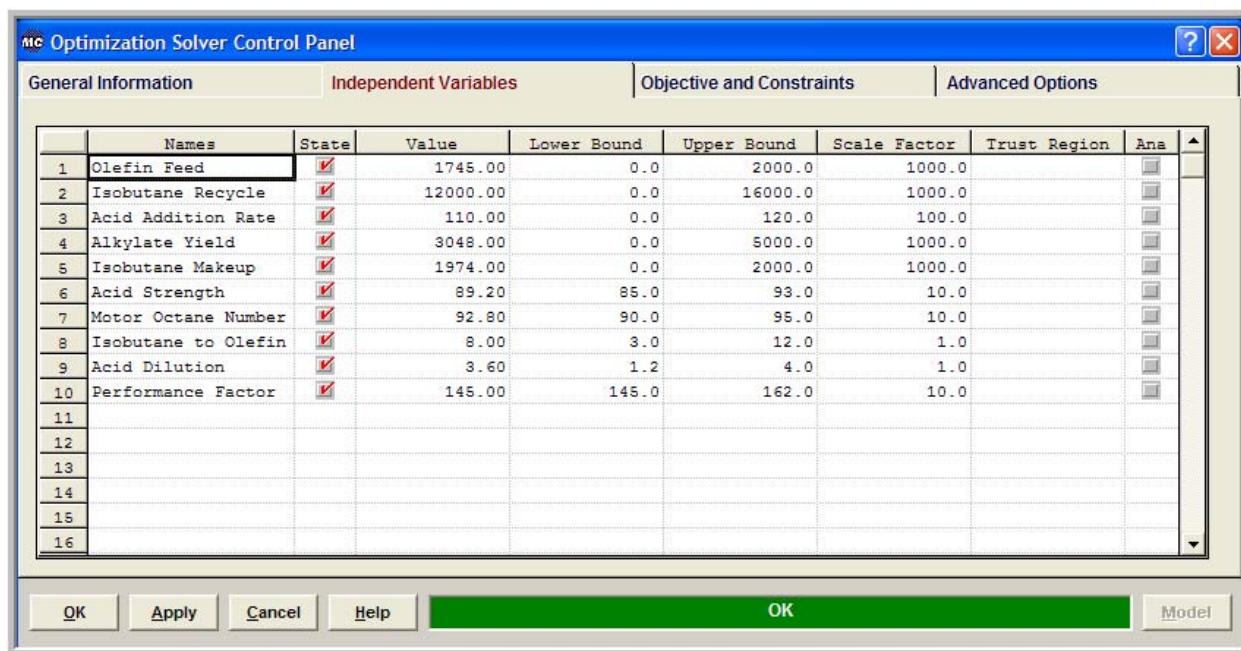
From the **Optimization Parameters** group enter:

- ❖ The number of Independent Variables (10)
- ❖ The number of Functions (Objective and Constraints) (8)

Optionally you may change the *Number of Iterations*, the *Optimality Tolerance*, the *Constraints Functions* or *Independent Variables Tolerance*, the *Print Level Control* Flag and the *Real and Integer Working Arrays* space requirements. From the **Optimization Problem Type** group choose *Non-Linear with General Constraints*, and from the **Optimization Solver Tasks** group choose *Maximize Objective Function*. You may also consider to *Check here to Test the Initial Values*. The test call to model is a very useful option since it allows you to see how good is your initial guess before you proceed with the optimization process. Finally you may consider entering a more appropriate value for the *Perturbation Step Size*. Keep in mind that if your functions are calculated down to machine precision, then a good value for the perturbation step size is the square root of the machine precision (this value is around 1.0E-8)

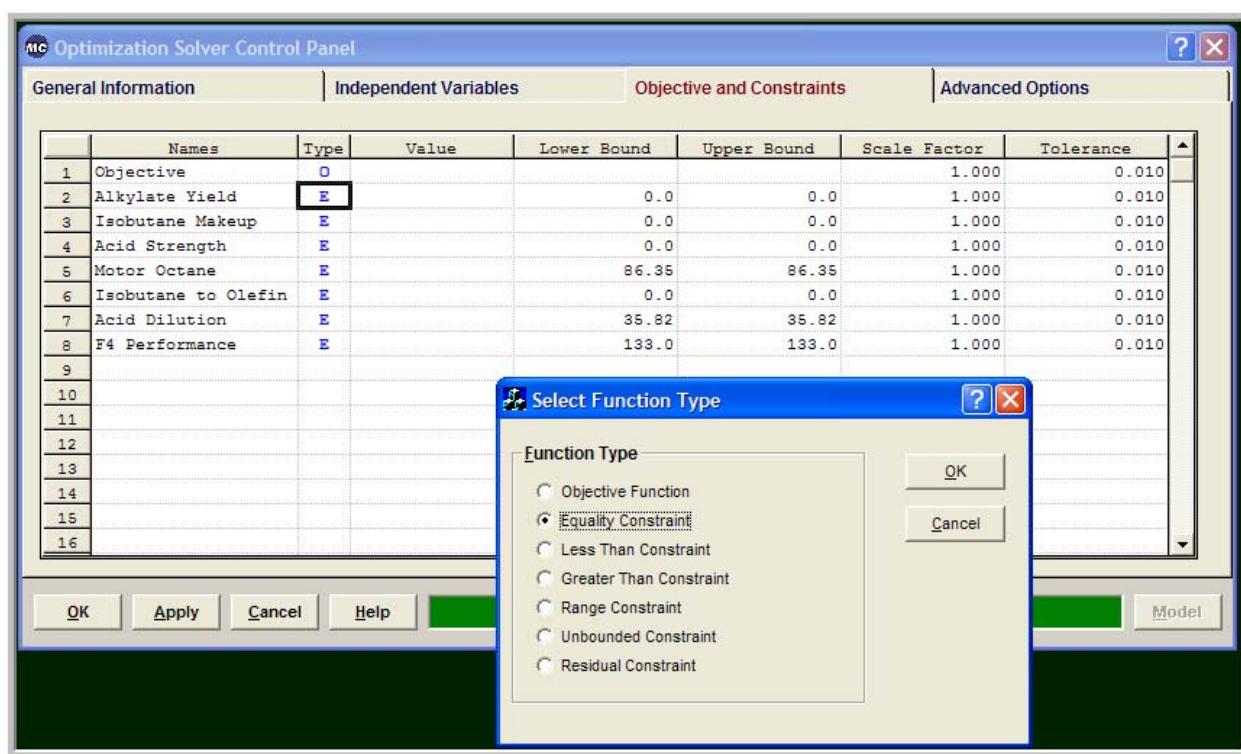
The Independent Variables

Next select the **Independent Variables** tab on the Optimization Solver Control Panel and enter the names of the variables you wish to optimize, their initial guesses and optionally their lower and upper bounds, the scale factor to improve the optimization and the size of the trust region. Notice that if a variable is not checked in the *State* column next to it, will remain fixed at its initial value during the optimization. The following data have been inserted for our example:



The Objective Function and Constraints

Next select the **Objective and Constraints** tab on the Optimization Solver Control Panel and enter the names and the types of the model functions. To enter the type perform a right mouse click in the *Type* column next the function you wish to specify and then click *Select Constraint*. The *Select Function Type* window will show up (see image below). Select the *Function Type* and then click **OK**. Next enter the lower and upper bounds for the constraints making sure that these bounds are identical for the equality constraints. For *Less Than* type constraints enter only the upper bound (the lower bound is taken as negative infinity). For *Greater Than* type constraints enter only the lower bound (the upper bound is taken as infinity). Optionally you may enter scale factors for the constraints and objective as well as overwrite the global tolerances for convergence by specifying an individual tolerance for each function. The following data have been inserted for our example:



Saving and Running

You are now ready to save your model and run it. New files are labeled **UNTITLED** until they are saved. Keep in mind that the maximum number of characters in a line is 132; the maximum number of lines in a file is infinity. In order to save your project:

- ❖ From the **File** menu, choose **Save**. This will save your model and create the Fortran code that will access the Nonlinear Optimization solver. The **Save As** dialog box appears.
- ❖ In the Directories box, double-click a directory where you want to store the source file (or down a directories path to the appropriate directory.)
- ❖ Type a filename (a filename cannot contain the following characters: \ / : * ? “ < > |) in the File Name box, then choose OK. The default extension given to a file is **AVW**.
- ❖ To view the Fortran code that you have just created from the **View** menu choose **Fortran Code**.

New files are labeled UNTITLED until they are saved. The maximum number of characters in a line is 132; the maximum number of lines in a file is infinity. Before you can save or close a window it must be active. To make a window active, either switch to the window (by clicking anywhere in it) or choose the window name or number from the Window menu.

You may now choose to compile, build and execute your project; to do that.

- ❖ From the **Build** menu choose **Compile** (or **Hit F2**)
- ❖ From the **Build** menu choose **Build EXE** (or **Hit F4**)
- ❖ From the **Build** menu choose **Execute** (or **Hit F5**)

Numerical Results

If everything goes well the results window will appear. In this window you can see the solution of your problem as well as various run statistics pertaining to the solution process:

```
Number of Independent Variables..... 10
Number of Dependent Variables..... 8
Number of User Specified Iterations..... 30
```

EXIT OPTIPLUS: SOLUTION FOUND

VARIABLE	STATE	LOWER BOUND	UPPER BOUND	INITIAL VALUE	FINAL VALUE
Olefin F	1 FR	0.000000E+00	2.000000E+03	1.745000E+03	1.917843E+03
Isobutan	1 FR	0.000000E+00	1.600000E+04	1.200000E+04	1.596152E+04
Acid Add	1 FR	0.000000E+00	1.200000E+02	1.100000E+02	4.359033E+01
Alkylate	1 FR	0.000000E+00	5.000000E+03	3.048000E+03	3.211347E+03
Isobutan	1 UL	0.000000E+00	2.000000E+03	1.974000E+03	2.000000E+03
Acid Str	1 LL	8.500000E+01	9.300000E+01	8.920000E+01	8.500000E+01
Motor Oc	1 UL	9.000000E+01	9.500000E+01	9.280000E+01	9.500000E+01
Isobutan	1 FR	3.000000E+00	1.200000E+01	8.000000E+00	9.365489E+00
Acid Dil	1 FR	1.200000E+00	4.000000E+00	3.600000E+00	2.076000E+00
Performa	1 FR	1.450000E+02	1.620000E+02	1.450000E+02	1.520000E+02
FUNCTION	STATE	LOWER BOUND	UPPER BOUND	INITIAL VALUE	FINAL VALUE
Objectiv	O	NONE	NONE	8.723872E+02	1.839425E+03
Alkylate	E OK	0.000000E+00	0.000000E+00	1.399924E+02	5.022278E-05
Isobutan	E OK	0.000000E+00	0.000000E+00	4.400000E-01	-3.601563E-13
Acid Str	E OK	0.000000E+00	0.000000E+00	1.077376E+04	-3.862624E-09
Motor Oc	E OK	8.635000E+01	8.635000E+01	8.419420E+01	8.635000E+01
Isobutan	E --	0.000000E+00	0.000000E+00	-1.400000E+01	1.891684E-02
Acid Dil	E OK	3.582000E+01	3.582000E+01	3.579000E+01	3.582000E+01
F4 Perfo	E OK	1.330000E+02	1.330000E+02	1.334000E+02	1.330000E+02
					LAGRANGE MULT

Number of Function Evaluations.....	21
Number of Optimization Iterations.....	21
Number of Line Search Evaluations.....	0

From the above results we see that the optimization process took 21 iterations. No line search was performed during this process; this means that the full Newton step was taken at every single iteration. It is instructive to see that the optimizer reports the Isobutane constraint as violated(--). This is due to scaling of the problem. Choosing the proper scaling factor can expedite the solution process and also lead to better results. The choice of scaling factors however, is an art, not a science, and sometimes it may lead to failure instead of success.