Andromeda™

Software for the Global Optimization of Multiple Functions with Constraints

User Guide

Table of Contents

What types of global optimization problems can Andromeda solve?

Functions (F's): The global optimization problems can involve a single function that is to be optimized, or multiple functions that are to be simultaneously co-optimized (also known as multiple objective optimization). For example:

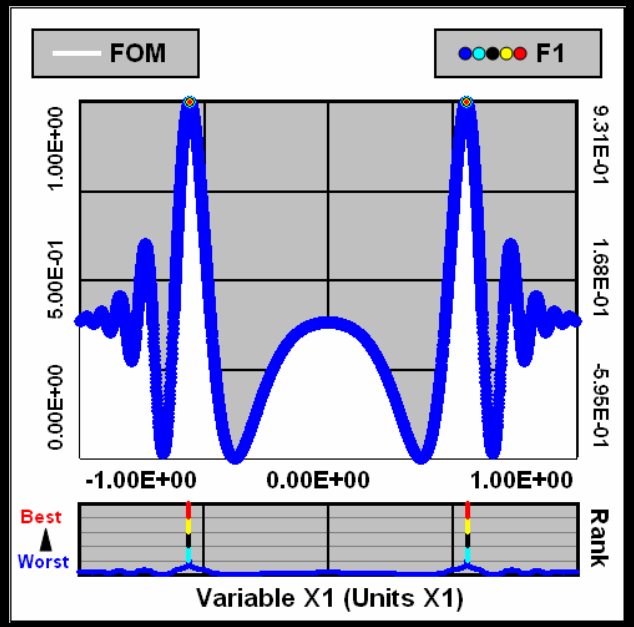
F1(X1,X2,…,Xn) = Maximum And F2 (X1,X2,…,Xn) = Minimum And F3 (X1,X2,…,Xn) = 0.0

The functions to be optimized can be continuous, or discontinuous, and even noisy. In its current implementation, the Andromeda software is an Excel application, and the functions to be optimized can be any valid expression in Excel (including logical functions, array formulas, user defined functions, and complex models that one links to either on another Excel worksheet or within another Excel workbook).

Constraints (C's): A constraint is any condition that a solution to an optimization problem must satisfy. The constraints specified by the user may be "Hard" (meaning that they must be exactly satisfied within the machine precision limits), or "Soft" (meaning that they are included as part of an objective function with some specified weight indicating relative importance), and they can also be in the form of either equality constraints or inequality constraints. Just as in the case of functions, any valid expression in Excel can be used to define each of the constraints.

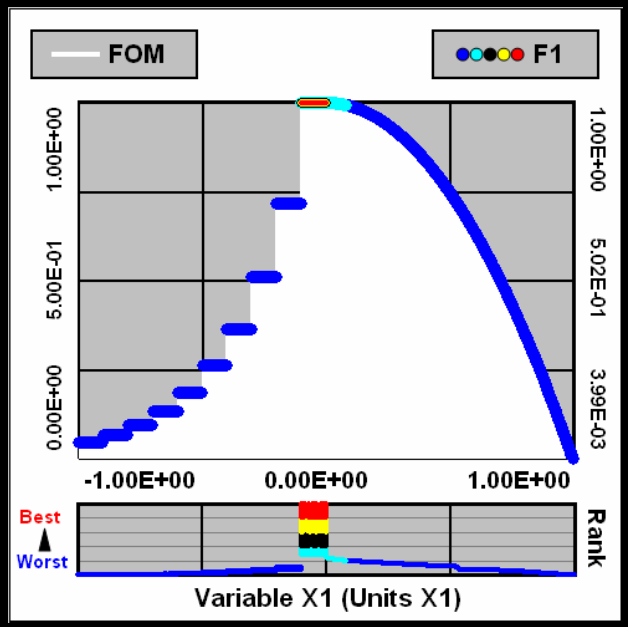
Variables (X's): The independent variables (X's) which are being solved for may be continuous or integer, and the search interval for each individual X may be either bounded or unbounded (in which case the particle distribution can evolve outside of the initial search region specified by the user).

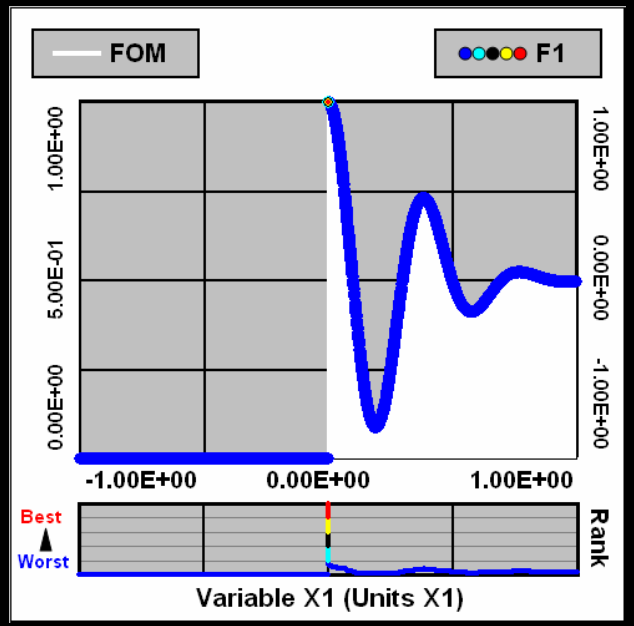
Some examples of global optimization problems Andromeda can solve



Multiple Global Optima

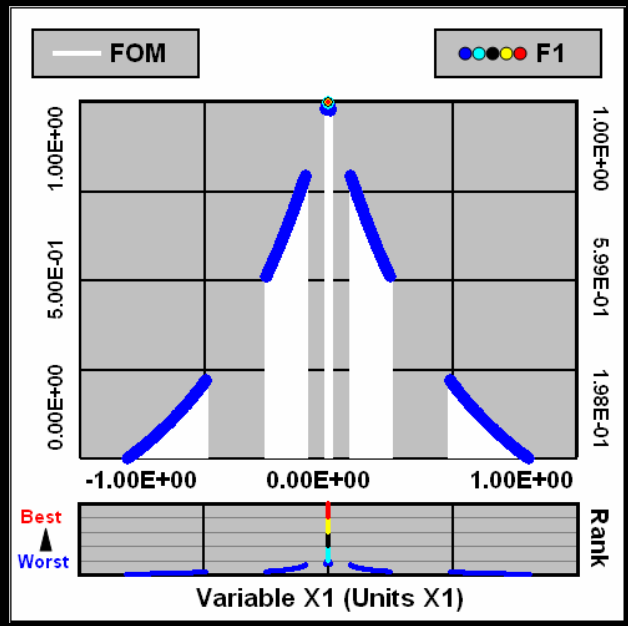
Continuous or Discrete Functions





- Discontinuity at Solution values
- Solutions on Constraint Boundaries
- Flat Regions of Function Space

Nonanalytic Constraints

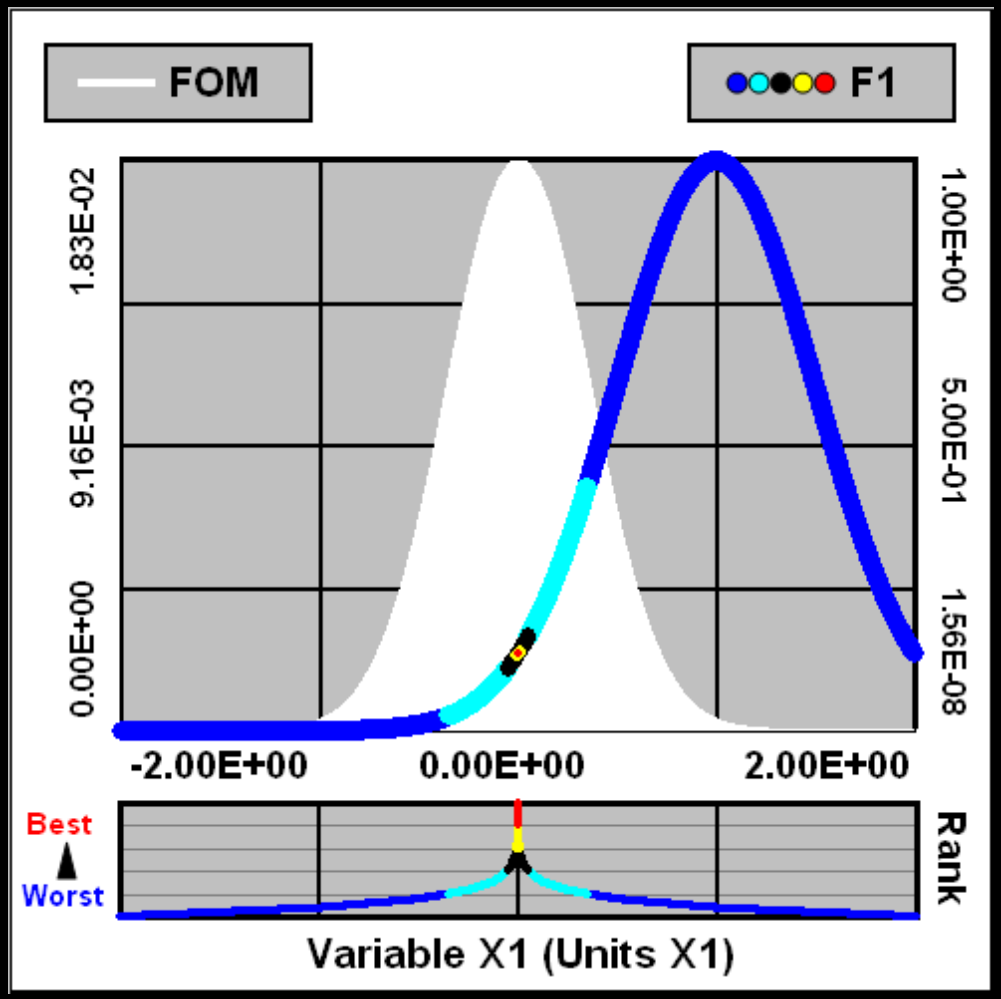


Some examples of global optimization problems Andromeda can solve

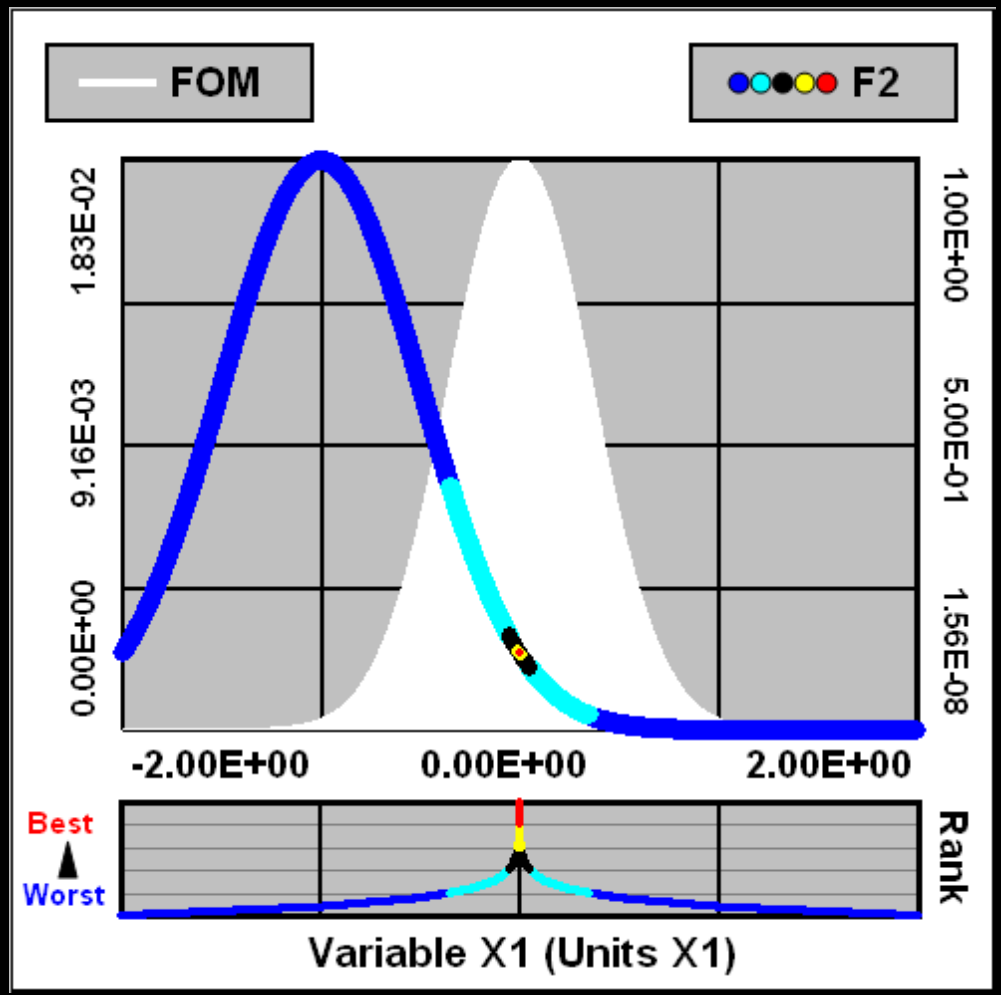
Multiple Objective Optimization

Example: Simultaneously Solve F1 = Maximum AND F2 = Maximum

F1 = Exp(-2*(X1-1)^2)



F2 = Exp(-2*(X1+1)^2)



Quick Start example

The user interface in Andromeda is highly intuitive, and anyone who is already familiar with Excel can begin solving problems immediately. The simple example presented in this section will provide the minimal amount of information that users need to begin using the software. A detailed description of the user interface and software functionality can be found in the later chapters of this user guide.

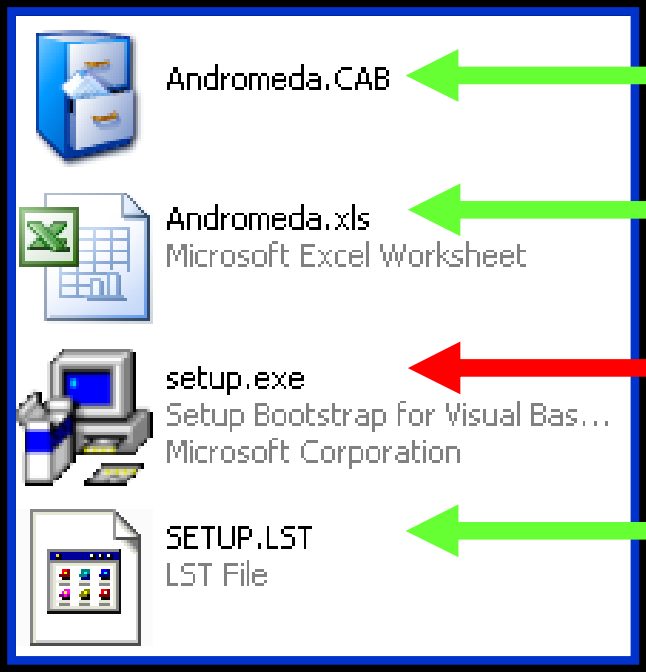
Andromeda.xls is an Excel Application which references a Dynamic Link Library (DLL) called Andromeda.dll. The DLL must be installed before one can run the Andromeda routines in the Excel application.

In this section we'll demonstrate:

- Installing the Andromeda DLL
- Setting up and solving a simple optimization problem
- Creating a data summary worksheet
- Creating a graphical summary worksheet

Installing the Andromeda DLL

The installation folder contains four files in addition to this User Guide:



A cabinet file containing the Andromeda DLL and support files.

The Andromeda application in Excel.

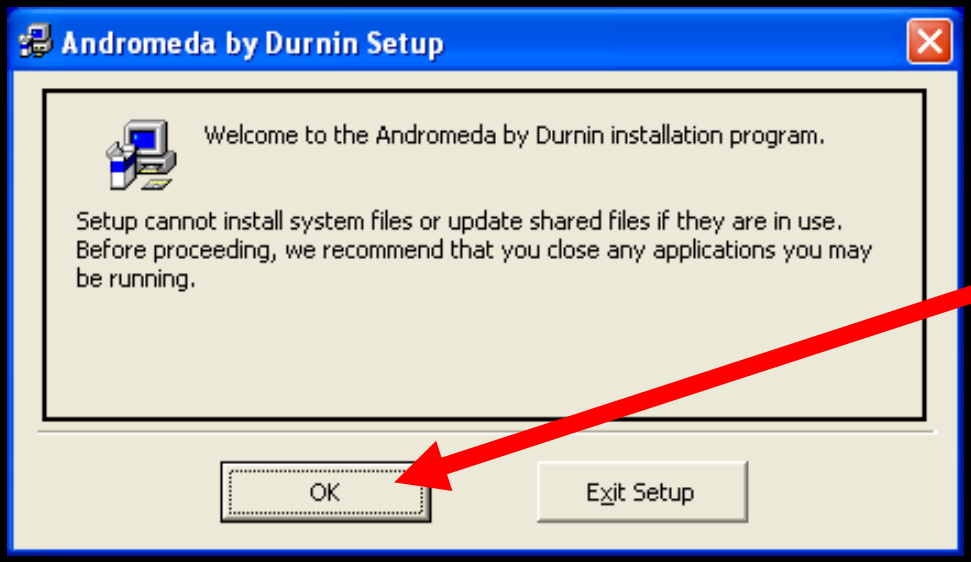
The executable file which will install the Andromeda DLL.

A Setup support file.

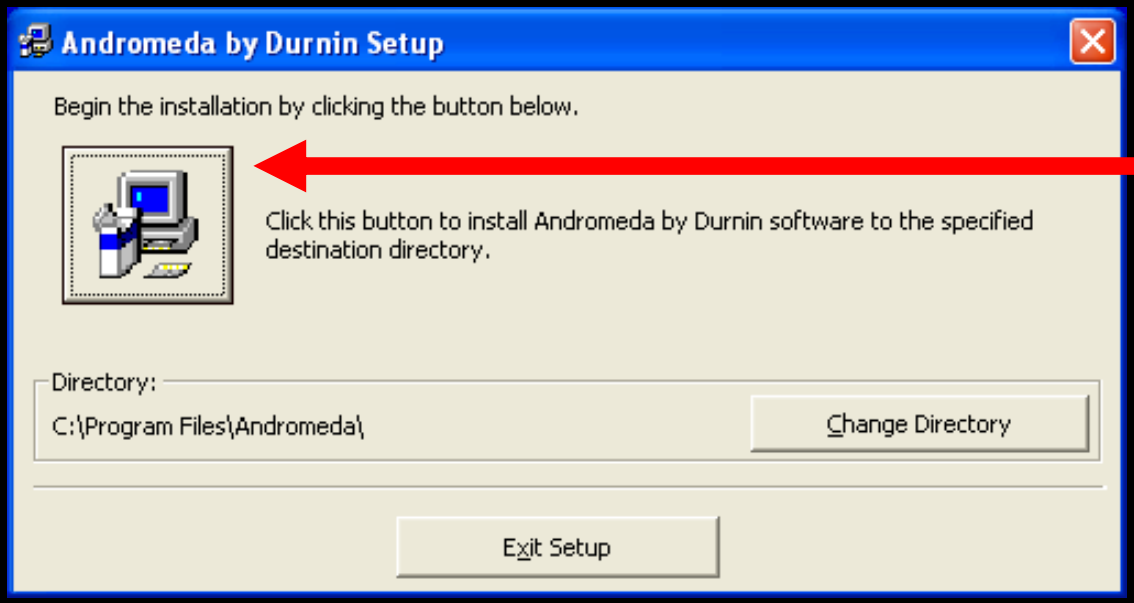
Copy the Andromeda.xls application to any location you like.

Double-click on the Setup.exe file to begin installing the Andromeda DLL.

Installing the Andromeda DLL



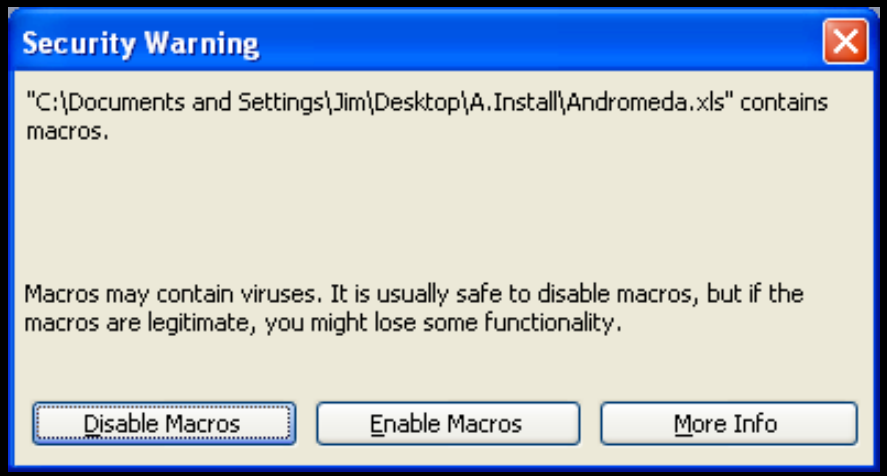
Click "OK"



Click this button to complete the installation of the Andromeda DLL

Opening the Excel Application (Andromeda.xls)

When you double-click on the Andromeda.xls file to open it, the following dialogue box will appear:



Click on "Enable Macros" in order to activate the executable code which references the Andromeda DLL and allows the algorithm to run.

If you are sharing your optimization results with someone who does not have a licensed Andromeda DLL, they should click on "Disable Macros" in order to open your file and work with it as a normal Excel workbook.

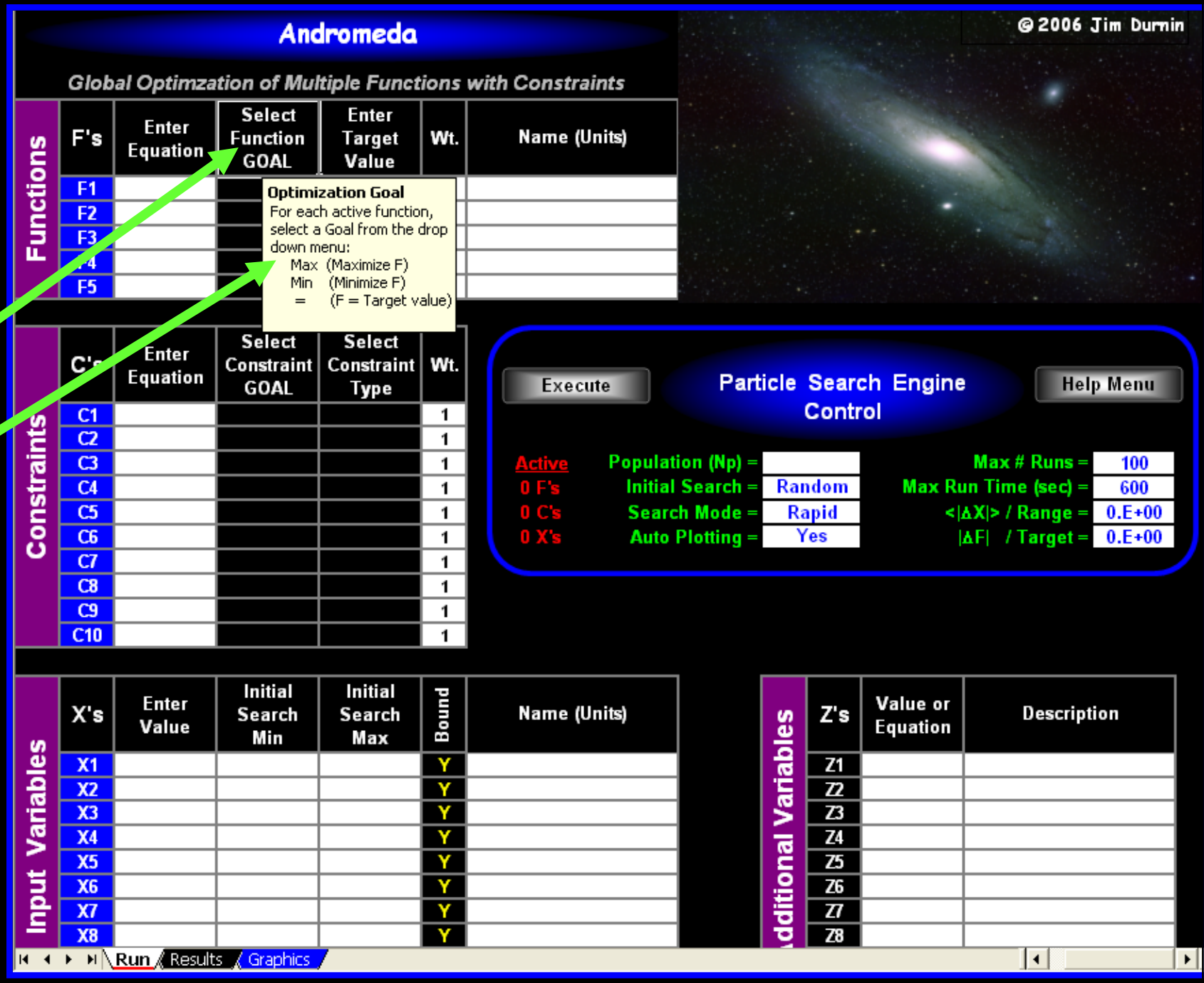
Inside the Excel Application (Andromeda.xls)

You will find 3 different worksheets with tabs labeled as follows:

"Run"	This is the worksheet on which the optimization problem is defined and program execution is controlled.
"Results"	This is the worksheet on which the results for the best solution are presented. A menu button on this worksheet allows the user to create a new data summary worksheet containing all of the solutions that were explored by the algorithm.
"Graphics"	This is the worksheet on which the user selects the types of graphical analysis that are of interest. A menu button on this worksheet allows the user to create a new plot summary worksheet containing every graph type that was selected.

Column Labels

On each of the worksheets, a single mouse click on any column label will pop-up a detailed description of that item.



Quick Start example: Solving for Phi

The Golden ratio, usually referred to simply as Phi, arises in the study of geometry, art, biology, the stock market, and many other subjects. It is a positive number which satisfies the equation:

$$\phi = 1 / (\phi - 1)$$

In this first example, we'll use Andromeda to solve the above equation for Phi by finding the value of X1 for which

$$F1(X1) = X1 - 1/(X1-1) = 0$$

on the interval $0 \leq X1 \leq 10$. Our solution for X1 will then be compared to the exact analytic result for Phi, which is given by the positive root of the quadratic equation:

$$\phi^2 - \phi - 1 = 0$$

$$\phi = [\, 1 + \sqrt{5} \,] / 2$$

Quick Start example: Solving for Phi

Activate and define function F1 on the "Run" worksheet

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1					
F2				1	
F3				1	
F4				1	

Activate F1 by clicking on its symbol. The background color will change from Blue to Red, indicating that F1 is now recognized as a function that is to be optimized.

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1	=x1-1/(x1-1)				
F2				1	
F3				1	
F4				1	

Type in the equation for F1:
" = x1 – 1/ (x1 – 1) "
and hit the enter key.
(A value of 1 will then appear since the initial value cell for X1 is currently blank and therefore assumed to be zero)

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1	1.00E+00			1	
F2		Max		1	
F3		Min		1	
F4		=		1	

Select the GOAL for F1
"="
from the drop down menu.

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1	1.00E+00	=	0.00E+00	1	Function1 (Units F1)
F2				1	
F3					
F4				1	

Enter the target value for F1
"0.0"
It's also good practice to enter a function name or description and units, if applicable.

Quick Start example: Solving for Phi

Activate and define input variable X1

Input Variables

X's	Enter Value	Initial Search Min	Initial Search Max	Bound	Name (Units)
X1	0.00E+00	0.00E+00	1.00E+01	Y	Variable1 (Units X1)
X2				Y	
X3				Y	
X4				Y	
X5				Y	
X6				Y	
X7				Y	
X8				Y	

- Activate X1 by clicking on its symbol.
- Enter an initial value for X1 (optional)
- Enter search range Minimum = 0
- Enter search range Maximum = 10
- Enter X1 name and units (optional)

Define Additional Variable Z1

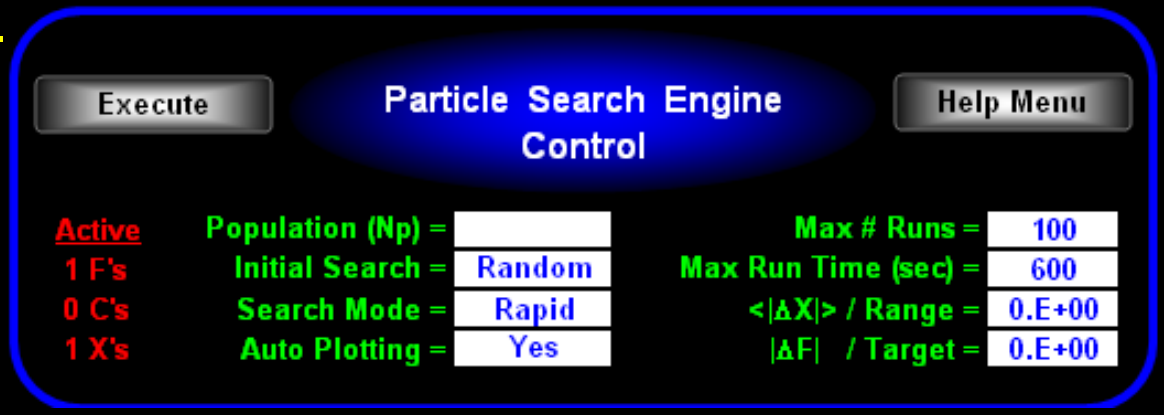
Additional Variables

Z's	Value or Equation	Description
Z1	=(1+SQRT(5))/2	Phi (Analytic Solution)
Z2		
Z3		
Z4		
Z5		
Z6		
Z7		
Z8		
Z9		

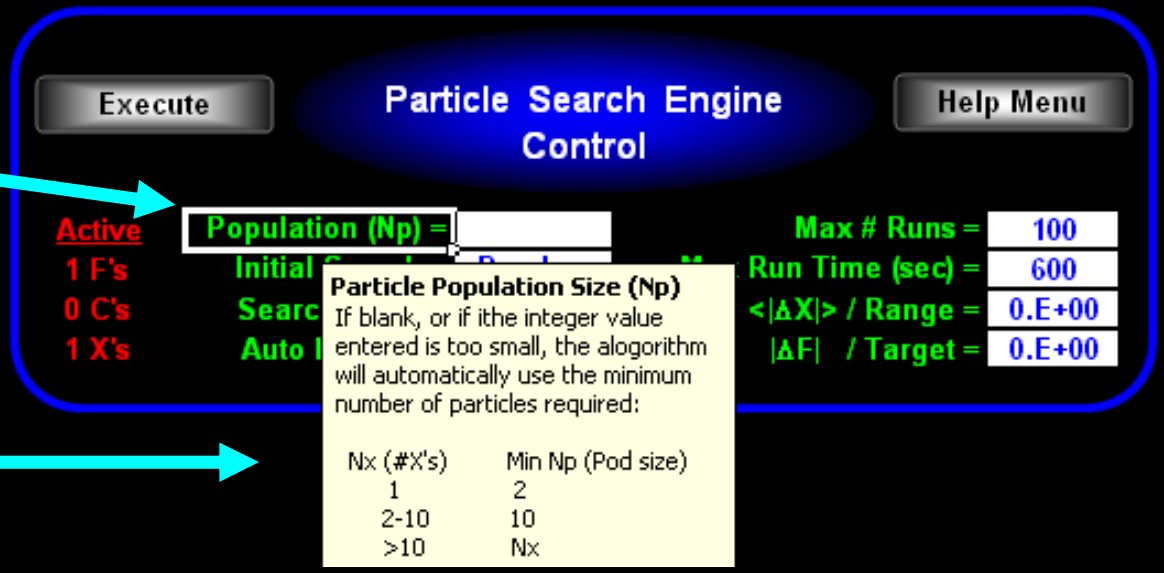
The Z's are additional variables that can be used for any purpose, such as defining fixed parameters or intermediate functions used to define the F's.

Quick Start example: Solving for Phi

The PSE Control Box contains all the algorithm parameters and program termination criteria that can be set by the user.



A single mouse click on the name of any control parameter will pop-up a detailed description of that item.



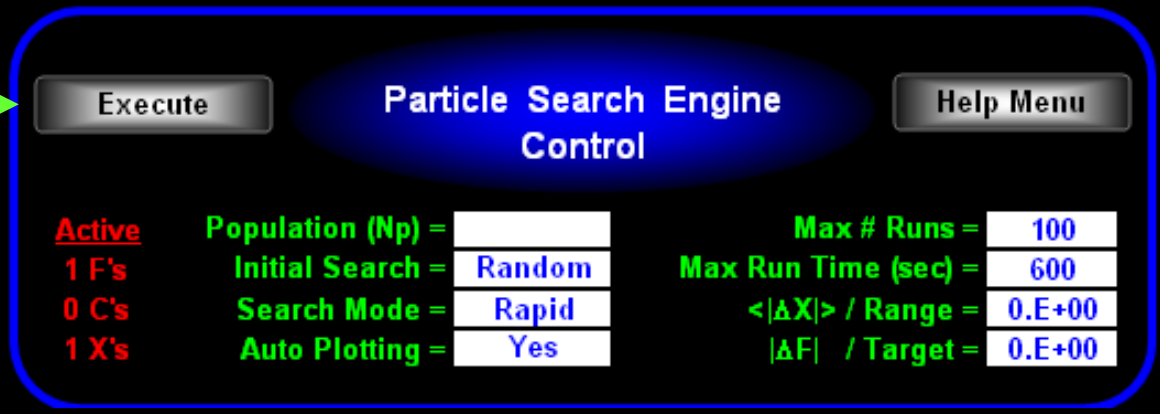
Population (Np)

The particle population (Np) represents the number of particles that are propagated in each iteration of the algorithm. In a one dimensional problem such as this, the minimum population required (also know as the "Pod" size, discussed in detail in Chapter II) is 2. By leaving this cell blank, the algorithm will automatically use the minimum Np = 2.

Quick Start example: Solving for Phi

Run the program !

Just click the "Execute" button located in the PSE Control Box.

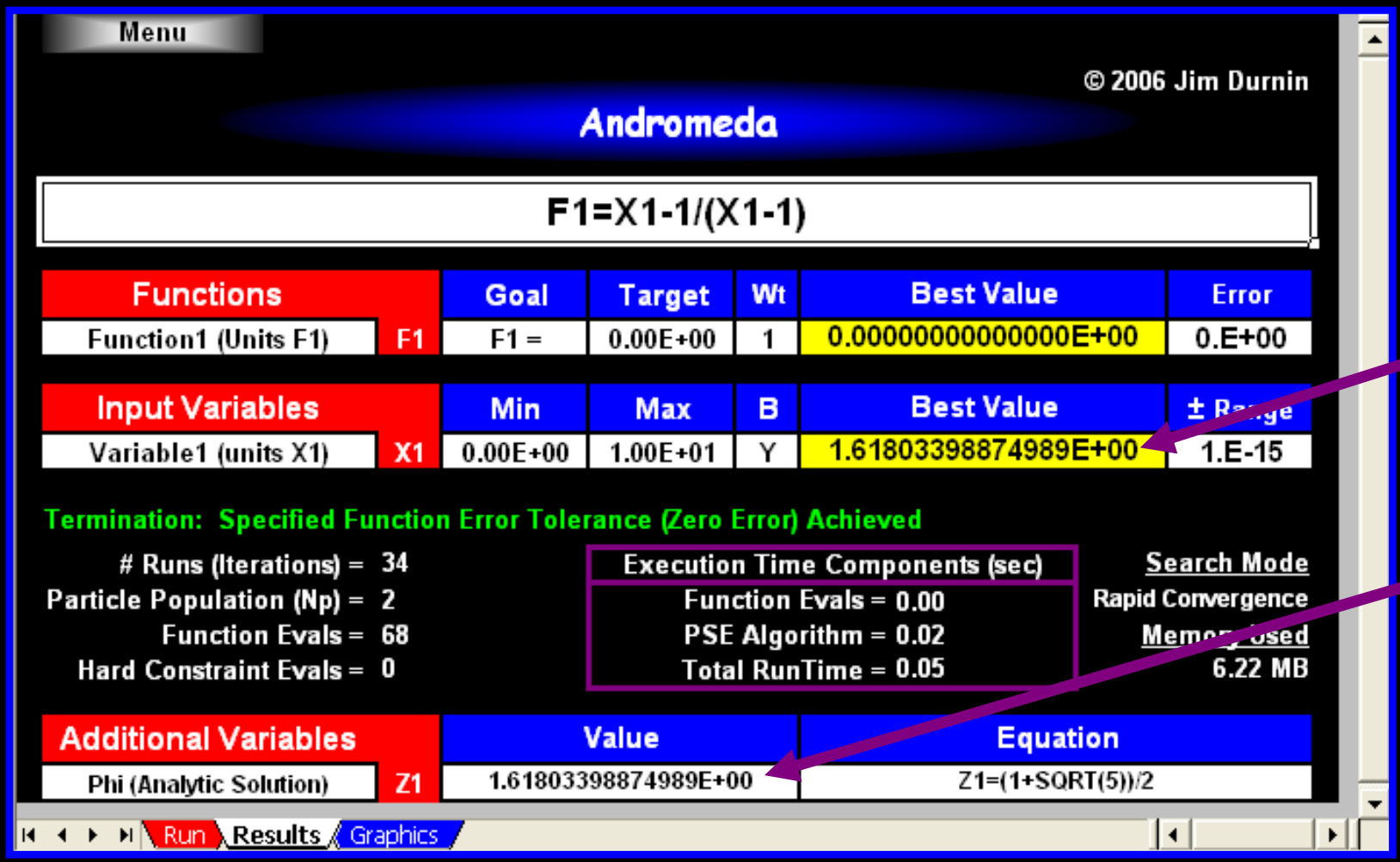


The status bar located at the bottom of the Excel window will display the progress while the PSE algorithm is running.

Status Bar

Quick Start example: Solving for Phi

The solution is reported on the "Results" worksheet.



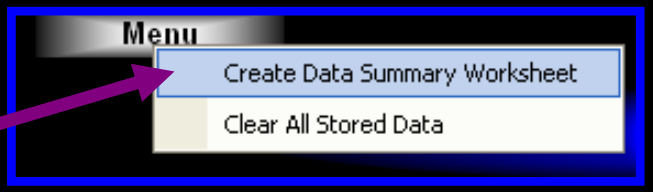
We see that our result for X1

matches the

analytic solution for Phi

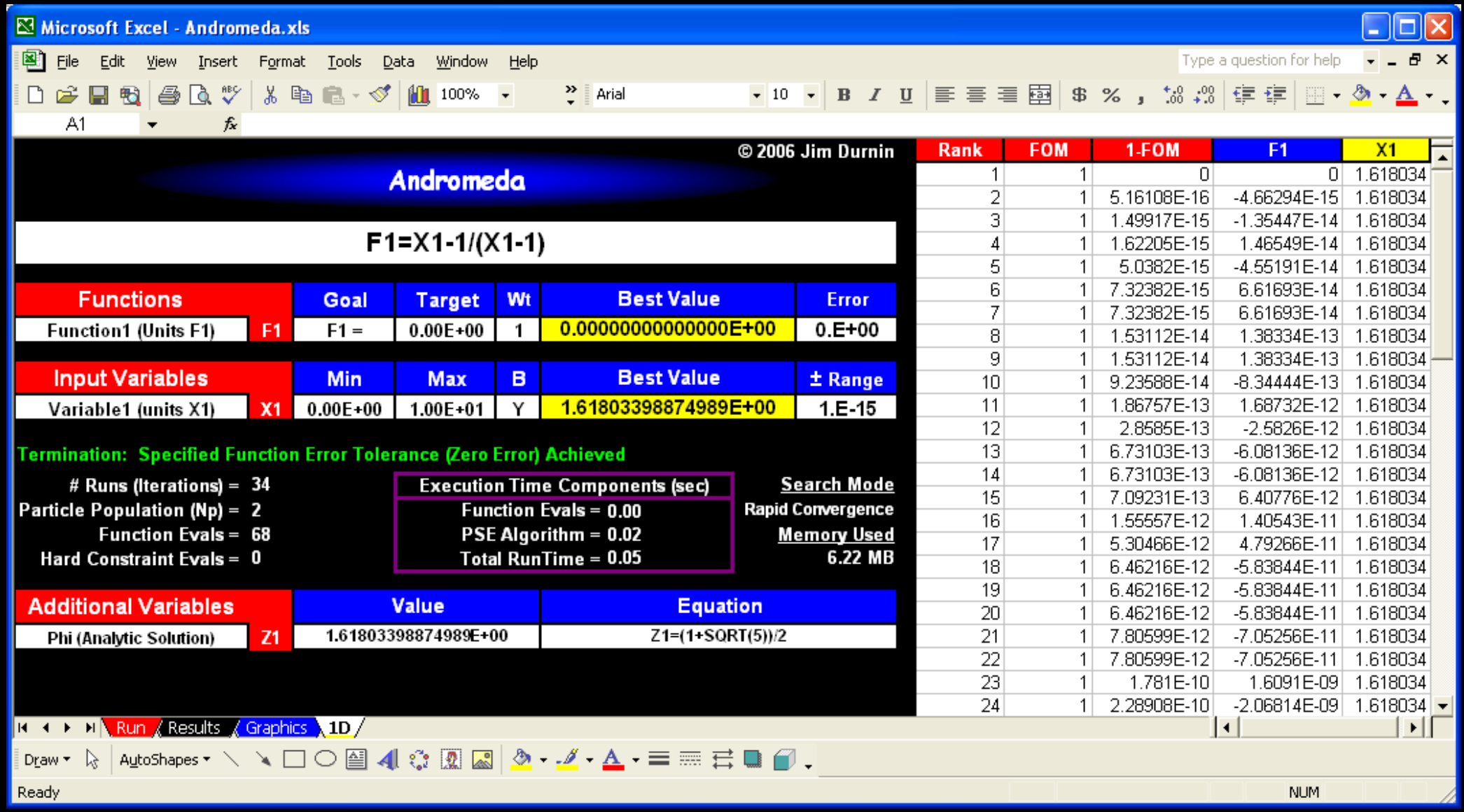
to 15 decimal places (the maximum precision of Excel).

To record all of the data that was generated, click on the Menu button and then select "Create Data Summary Worksheet."



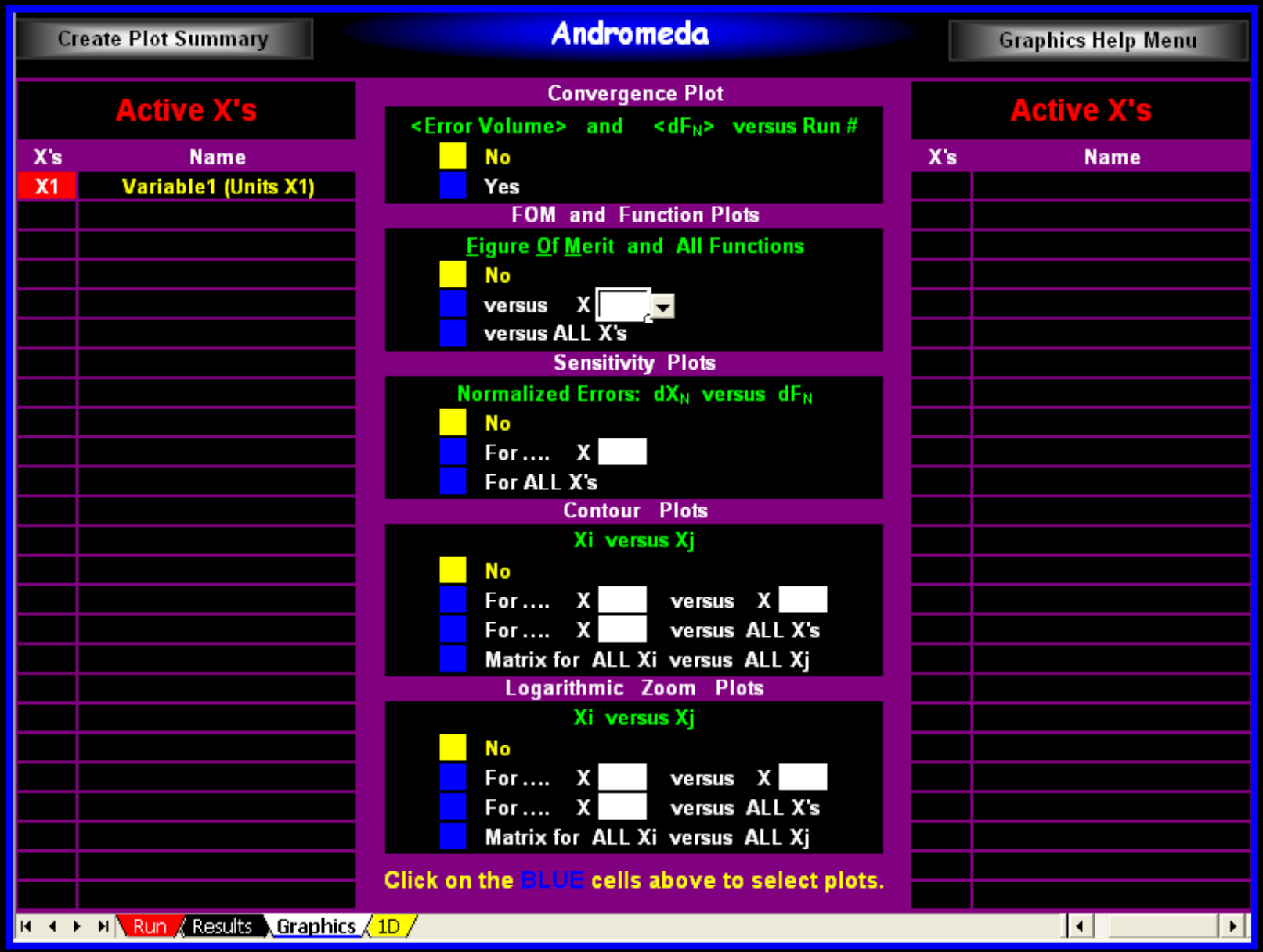
Quick Start example: Solving for Phi

A new worksheet named "1D" (D for Data) is then created which contains the results summary and ALL of the function evaluations (rank ordered from best to worst).



Quick Start example: Solving for Phi

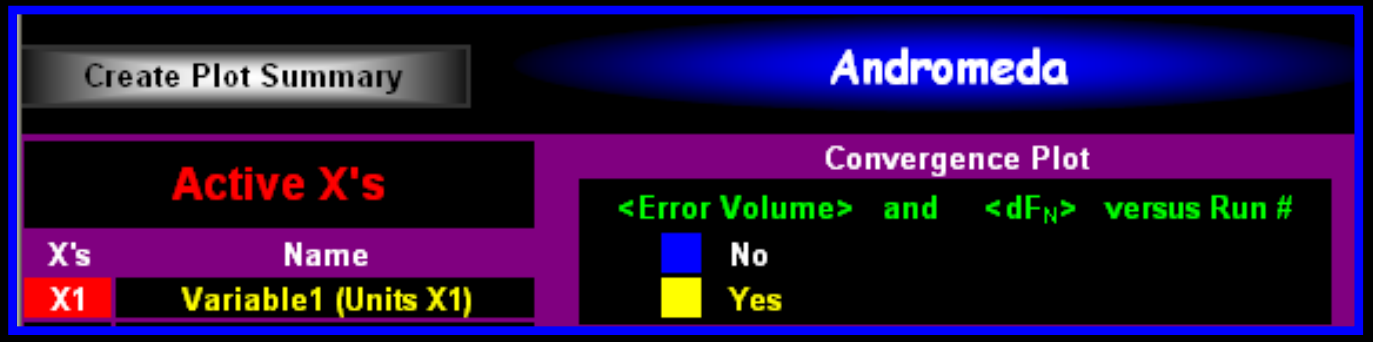
Now we'll demonstrate how to create a plot. Click on the "Graphics" worksheet tab.



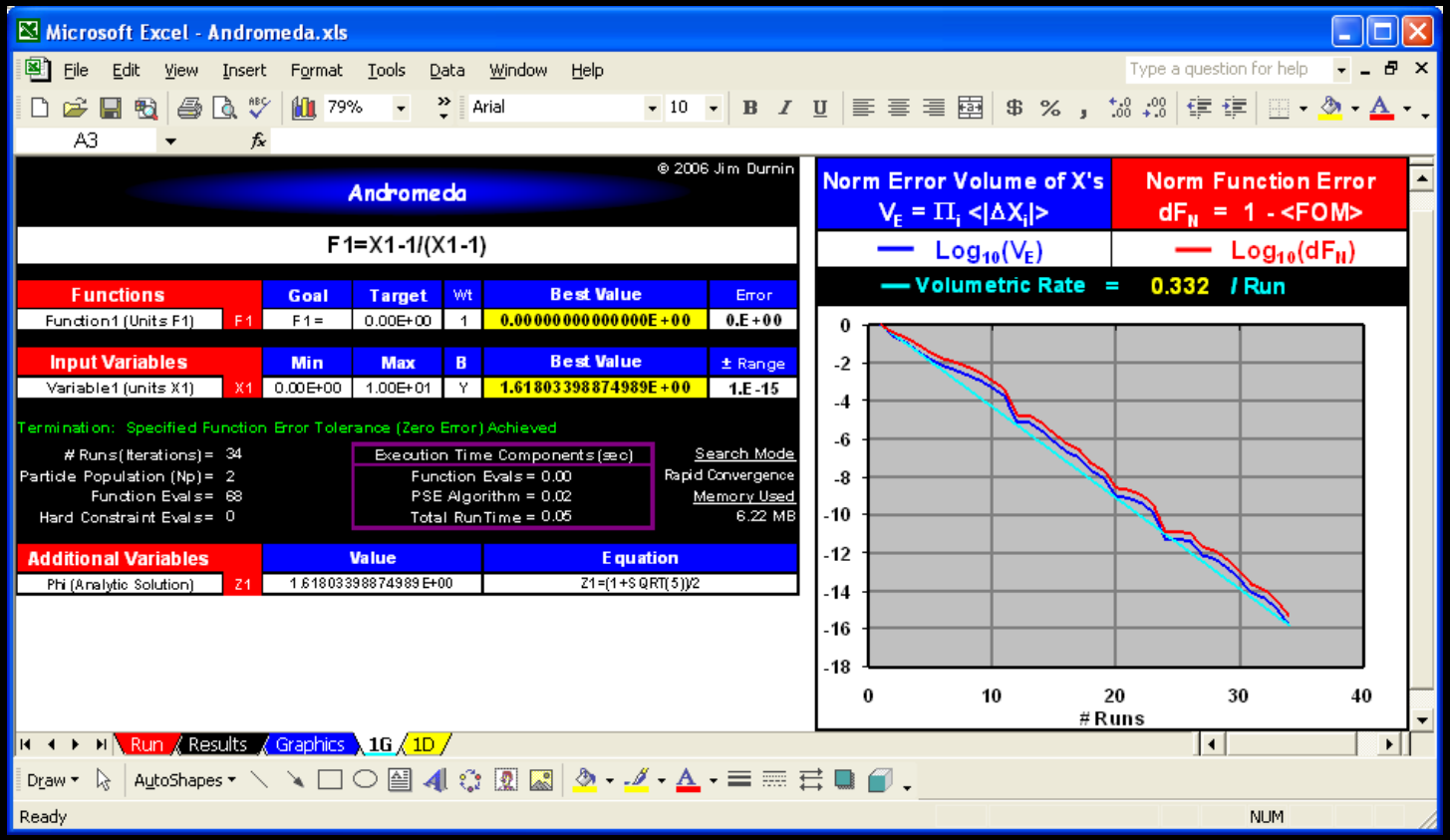
Quick Start example: Solving for Phi

Click on "Yes" to select the Convergence Plot.

Then click the "Create Plot Summary" button.



A new worksheet named "1G" (G for Graphics) containing the results summary and the Convergence Plot will then be added to the workbook.



The Convergence Plot shows a Volumetric Rate of 0.332, which is the factor by which the error in X1 is reduced (on average) in each iteration.

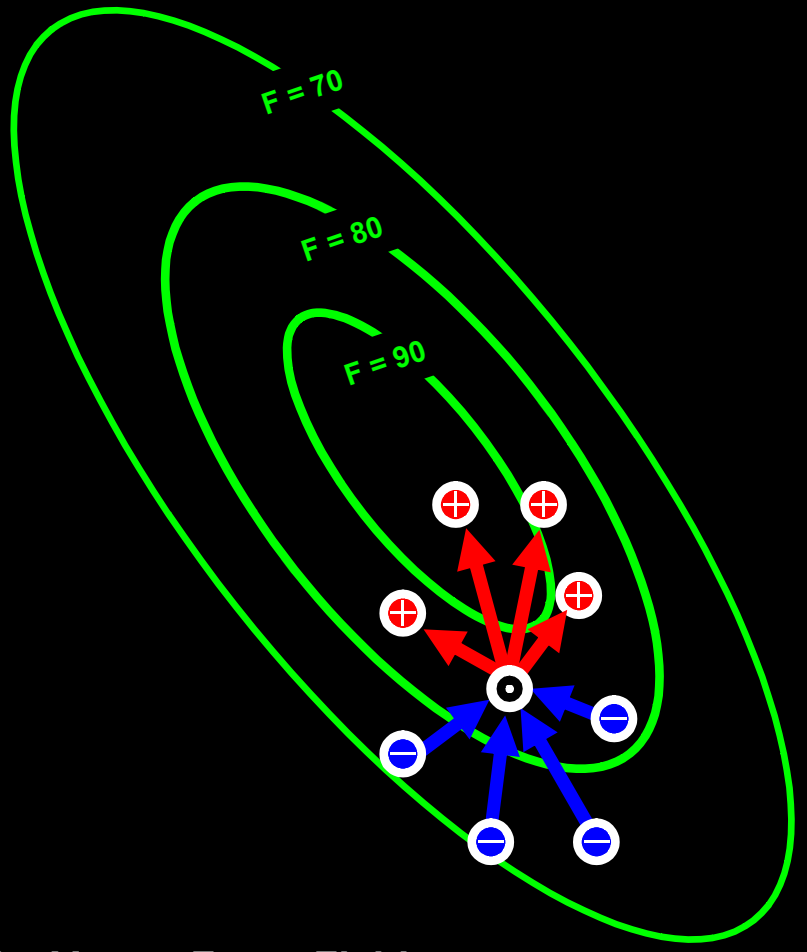
The particle population in this example is 2, so there are 2 function evaluations made in each iteration.

How it works

In the PSE algorithm, each sample of the function(s) at some particular coordinates within the search space is conceptually viewed as a "particle." The initial particle positions can be chosen either randomly, or as an array that uniformly spans the search space. The particle distribution then evolves as each particle moves towards increasingly better functional values with each iteration of the algorithm.

Particle Propagation Direction: The direction of propagation for each particle is governed by a Vector Force Field which is created by the nearest neighbor particles. Each particle is attracted to the nearest neighbor particles that have "better" functional values, and is repelled by the nearest neighbor particles having "worse" functional values.

Particle Propagation Distance: The particle propagation distance for each iteration is governed by simple rules involving factors such as the local particle density and some aspects of the general behavior of the function.



The Vector Force Field

Each propagating particle: ⊙

is attracted to particles with improved values: ⊕

and repelled by particles with worse values: ⊖

A new and universally applicable method for global optimization

The exact details of the PSE algorithm are proprietary, but in essence the vector force field governing particle motion causes each particle to move along a direction that is roughly the local path of steepest ascent for that particle, much like making an approximate gradient calculation. Unlike the case of an actual gradient calculation, however, the functions that are to be optimized do NOT need to be smooth and differentiable, or even continuous when using the PSE algorithm.

In addition, the particles are intelligent in the sense that they follow simple rules which determine the propagation distances chosen for each move. These rules make use of both local and global information that is shared by all of the particles.

The particles
move locally but "think" globally.

The PSE algorithm is an evolutionary algorithm in which the particle distribution continuously evolves towards the best solutions. Unlike other evolutionary techniques for finding the global optima of nonsmooth functions, the PSE algorithm can provide rapid and reliable convergence to EXACT solutions (within the precision limits of the machine, or to the precision specified by the user).

Comparison to other methods of function optimization

Method of Steepest Ascent: A very broad class of algorithms use gradient calculations in order to search for the local optimum that is closest to the initial starting point specified by the user. These algorithms can often have extremely rapid convergence to local optima, especially in problems that they are specifically designed for (such as functions that are polynomials of some particular finite order). The key drawback to such methods when solving general optimization problems is that the functions to be optimized must be continuous and smooth (having first and sometimes also second derivatives) in order for these methods to be employed.

Evolutionary Algorithms: Global optimization requires the ability to explore all of the local optima within the designated search space, and this can be accomplished with the use of evolutionary algorithms such as the Genetic Algorithm or Particle Swarm Optimization. The Genetic Algorithm is the method usually chosen to handle nonsmooth functions. The key drawback of these evolutionary techniques is that the local convergence is usually extremely slow, far too slow to make high precision solutions feasible.

The PSE algorithm works with any type of function and has a convergence rate that is many orders of magnitude faster than evolutionary methods such as the Genetic Algorithm. This is achieved by propagating the particles approximately along their local paths of steepest ascent without requiring an actual gradient calculation, thereby combining the best aspects of steepest ascent and evolutionary techniques.

The PSE algorithm parameters

There are three parameters of the PSE algorithm that are specified by the user:

- Particle Population Size (Np)
- Initial Search
- Search Mode

Minimum Population Size (Pod Size)

The user specifies the particle population size (Np) representing the number of particles that will propagate in each iteration of the algorithm. The minimum number of particles that must be used in each iteration of the PSE algorithm will be referred to as the "Pod" size. For a given number of dimensions in the function space, Nx = # independent variables (X's) that we are solving for, the corresponding Pod sizes required in the PSE algorithm are:

Nx	Pod Size
1	2
2-10	10
≥ 10	Nx

The Pod size also represents the number of nearest neighbor particles that are involved in propagating a particle.

Particle Population Size (Np)

What Particle Population Size should be used to solve an optimization problem?

- If the problem has only a single optimum and no other local optima within the search space, then the exact solution can be obtained by using the minimum number of particles:

Np = Pod size

(Note: If the input value for Np is left blank, or if a value smaller than the Pod size is entered, the algorithm will automatically set Np equal to the Pod size)

- If the problem has many local optima, then using a sufficiently large population of particles will enable a massively parallel search of the function space, with the particles simultaneously converging on multiple local optima until the global optimum dominates. As a rule of thumb, one should choose a population size of at least:

Np > (Pod Size) x (# Local Optima)

When solving an optimization problem, the number of local optima is generally not known a priori, and a good approach is to run a large number of particles for a short number of iterations, and then do a contour matrix plot analysis to see how many local optima exist.

Initial Search

There are two options for selecting the particle locations in the initial population of Np particles, and the "Array" option is usually the best choice because it provides the most efficient initial search of the function space.

Random: The Np values for each X variable being solved for are chosen randomly (with uniform distribution) between the Max and Min values specified by the user for the initial search range of each X.

Array: The Np values for each X variable being solved for are chosen to form a spatial array that uniformly spans the hypercube defined by the Max and Min values specified by the user for the initial search range of each X. The array values have a small random component (on the order of one-tenth the lattice spacing). Defining Nx as the number of X's, the algorithm solves for the largest integer value M for which:

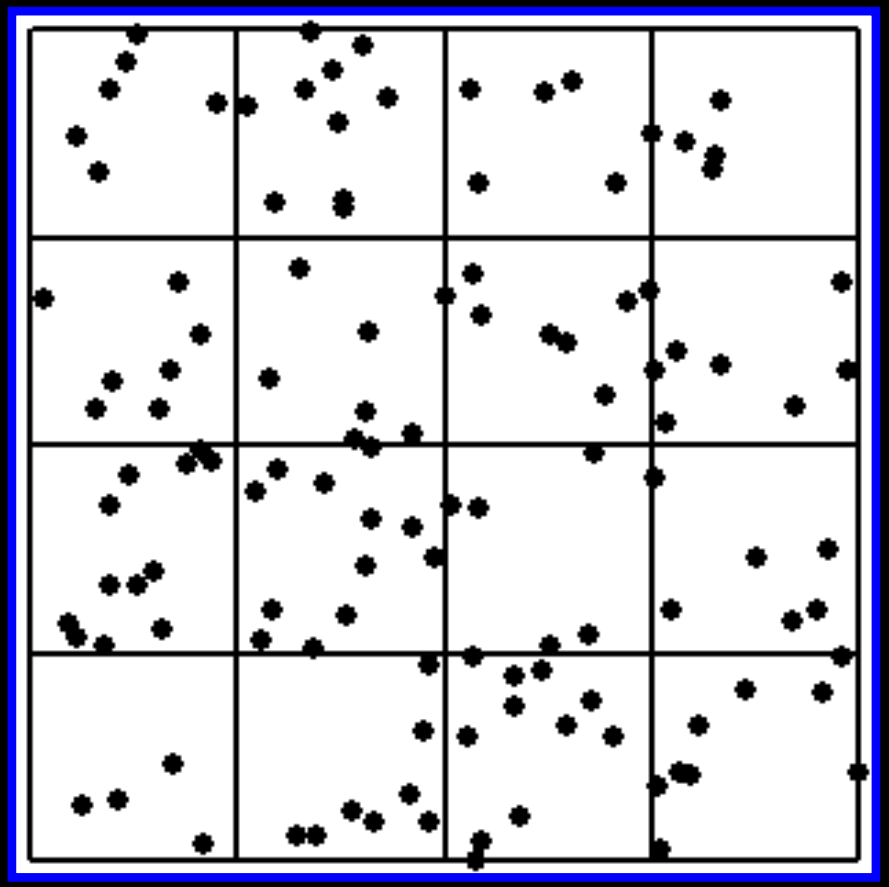
$$(M)^{Nx} \leq Np$$

If M = 1, the number of particles Np is too small to create an array in the Nx dimensional space, and all positions are assigned randomly as described above.

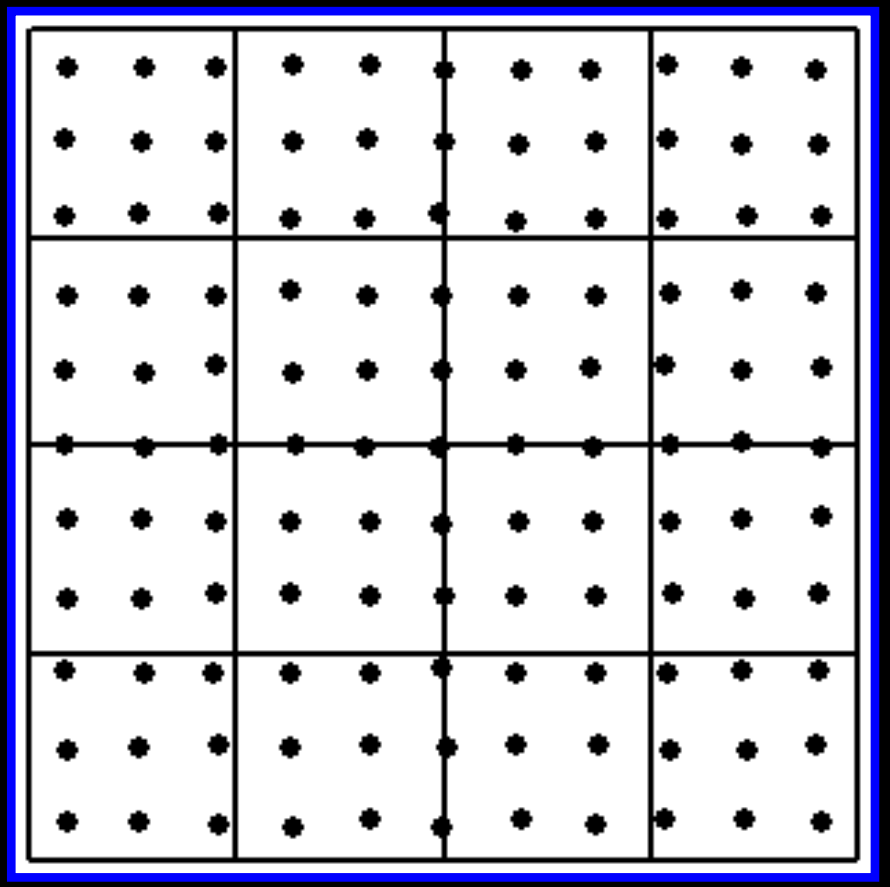
If M > 1, then (M)^Nx particles are assigned to the array positions, and the remainder of the Np particles (if any) are assigned to random locations.

Initial Search Example

Here's an example of the initial particle distributions produced in a two dimensional problem (Nx = 2) when the particle population size is Np = 121 = $(11)^2$.



"Random"



"Array"

(Notice that a small random variation is added to the array positions.)

PSE Search Modes

The Andromeda application provides two different search modes for the user to choose from. In both cases, the initial population of Np particles is chosen either randomly, or as an array that uniformly spans the search space, whichever the user specifies. Both modes then proceed to search for global optima, and the only difference lies in the selection rules for choosing the new population of particles that propagate in the next iteration.

Search Mode 1: Most Rapid Convergence

In this mode, after each of the particles in the current population has propagated (thus creating Np new particles), the best Np of those 2Np particles are chosen to be the new population that propagates in the next iteration. This selection rule provides the most rapid convergence possible since only the best particles are always propagated in every iteration. If many roughly equivalent local optima are present, however, this search mode is not as thorough in finding global optima as the second search mode described below.

Search Mode 2: Most Thorough Search

In this mode, after each particle propagates and creates a new particle, the new particle is chosen for the next population if it has improved relative to the previous particle; otherwise, the previous particle is chosen and propagated again in the next population. The result of this selection rule is that the particles will never cease exploring any local optimum once convergence has begun, and therefore this mode provides the most thorough search of local optima that is possible. As one would expect, the convergence is generally slower than the search mode above since we are not always choosing the globally best particles for each new population that propagates.

The "Run" Worksheet (for defining the optimization problem to be solved)

Andromeda

Global Optimzation of Multiple Functions with Constraints

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1	1.00E+00	=	1.00E+00	1	Function1 (Units 1)
F2	-7.13E-18	Max	0.00E+00	1	Function2 (Units 2)
F3	-5.74E-49	Min	0.00E+00	1	Function3 (Units 3)
F4				1	
F5				1	

Constraints

C's	Enter Equation	Select Constraint GOAL	Select Constraint Type	Wt.
C1	-4.51E-17	= 0	Soft	1
C2	-3.80E-17	> 0	Hard	1
C3	-8.31E-17	= 0	Hard	1
C4				1
C5				1
C6				1
C7				1
C8				1
C9				1
C10				1

Execute

Particle Search Engine Control

Help Menu

Active
3 F's
3 C's
8 X's

Population (Np) =	10	Max # Runs =	100
Initial Search =	Random	Max Run Time (sec) =	120
Search Mode =	Thorough	<|ΔX|> / Range =	0.E+00
Auto Plotting =	Yes	|ΔF| / Target =	0.E+00

Input Variables

X's	Enter Value	Initial Search Min	Initial Search Max	Bound	Name (Units)
X1	-4.51E-17	-1.00E+00	1.00E+00	Y	Variable X1 (Units X1)
X2	-3.80E-17	-1.00E+00	1.00E+00	Y	Variable X2 (Units X2)
X3	-6.50E-09	-1.00E+00	1.00E+00	Y	Variable X3 (Units X3)
X4	2.01E-08	-1.00E+00	1.00E+00	Y	Variable X4 (Units X4)
X5	-8.98E-10	-1.00E+00	1.00E+00	Y	Variable X5 (Units X5)
X6	1.83E-09	-1.00E+00	1.00E+00	Y	Variable X6 (Units X6)
X7	8.09E-09	-1.00E+00	1.00E+00	Y	Variable X7 (Units X7)
X8	-1.44E-09	-1.00E+00	1.00E+00	Y	Variable X8 (Units X8)
X9				Y	
X10				Y	

Additional Variables

Z's	Value or Equation	Description
Z1		
Z2		
Z3		
Z4		
Z5		
Z6		
Z7		
Z8		
Z9		
Z10		

The PSE algorithm can be applied to optimization problems having any number of Functions (F's), Constraints (C's), and Variables (X's).

Version 1.0 of the Andromeda software in Excel implements the following:
5 F's
10 C's
50 X's

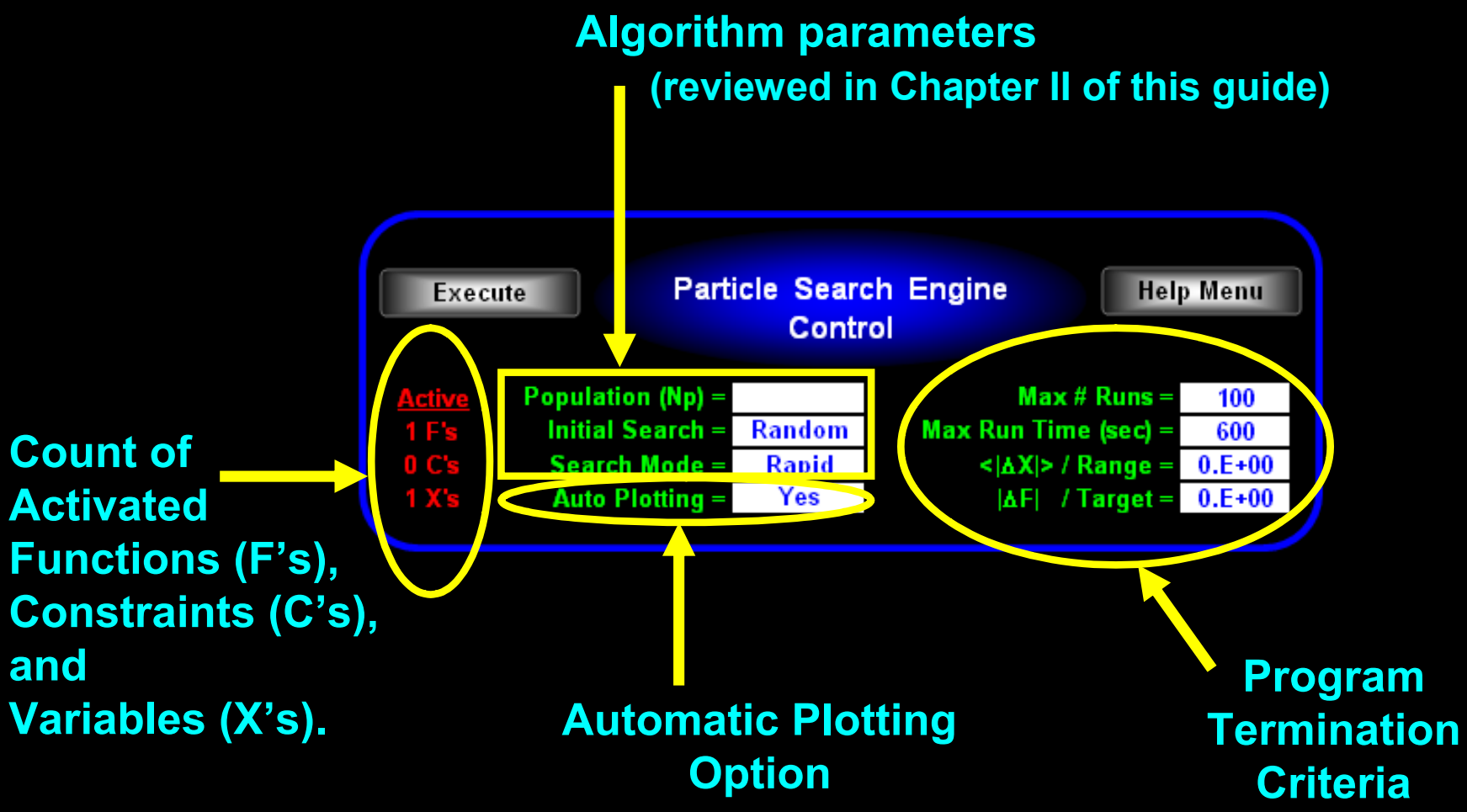
The Functions, Constraints, and Variables are activated and deactivated by clicking on the corresponding symbols.

Andromeda Version and License Information

A single mouse click on the Andromeda Logo located at the top of any worksheet

Will pop up a dialogue box with the Version and License information.

The PSE Control Box



Note: A single mouse click on the name of any parameter will pop-up an information box containing a detailed description of that item.

Program Termination Criteria

Max # Runs (Required Input)

Enter an integer number representing the number of iterations of the PSE algorithm to be performed. This will also determine the amount of memory to be reserved by the program.

Max Run Time (Optional Input)

Enter the maximum run time in seconds. If left blank, this termination criteria is not used.

< |ΔX| > / Range (Optional Input)

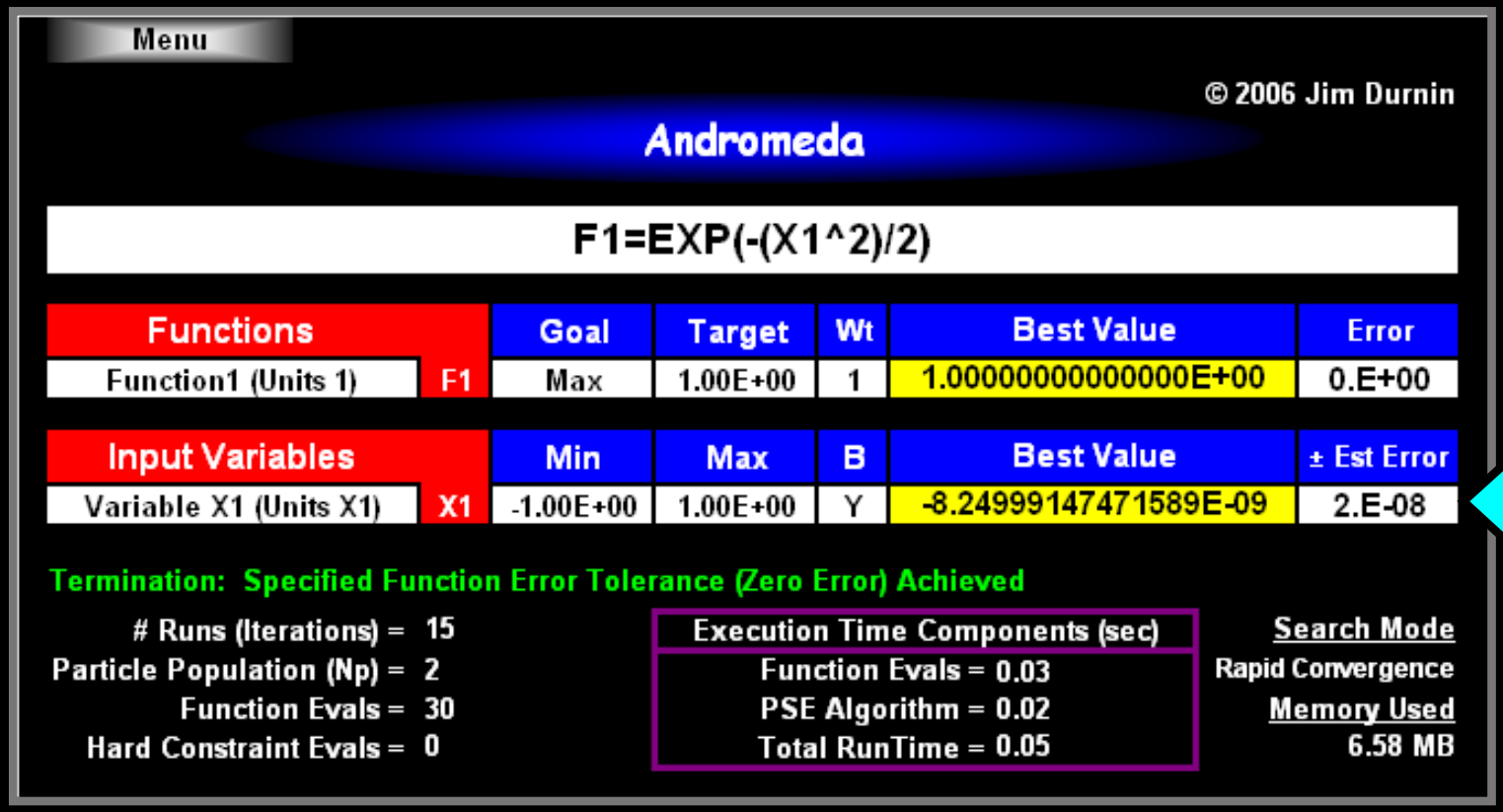
Enter any decimal value greater than or equal to zero. Program execution will terminate when, for the best Pod of particles, the average range of values for each X relative to the initial search range for each X is less than or equal to this number. If left blank, this termination criteria is not used.

|ΔF| / Range (Optional Input)

Enter any decimal value greater than or equal to zero. Program execution will terminate when, for the best Pod of particles, the relative error for each Function value is less than or equal to this number. Whenever a Target value is zero, the absolute error is used instead of the relative error. If left blank, this termination criteria is not used.

Note: Whenever the program is running, execution can always be terminated by hitting the Escape key (Esc), but the data generated will be lost.

The "Results" Worksheet

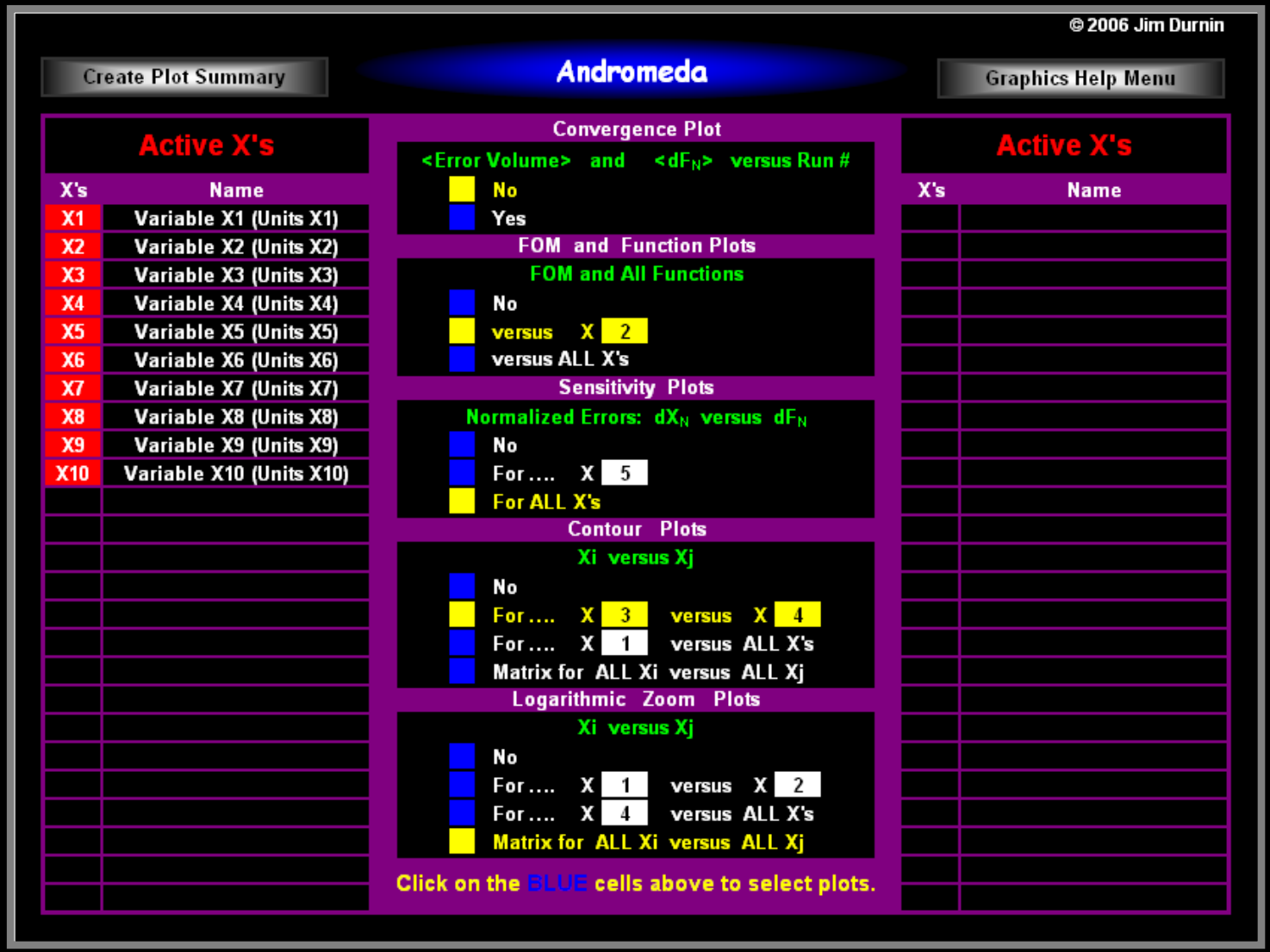


A novel aspect of the PSE algorithm is that the particle density provides an accurate estimate of the Error in the solution values at every step of the particle evolution.

One of the options provided by the Menu button on the Results worksheet will create a new worksheet summarizing all of the function and corresponding variable values (rank ordered from best to worst) obtained as the particles converged to the best solution. The user can then analyze all of the data in any manner desired. Many graphical analysis tools are included in the Andromeda application, and are described in the next section.

The "Graphics" Worksheet

A key benefit of the PSE algorithm is that as the particles converge on solutions, they leave behind a trail, the graphical analysis of which can give the user a great deal of insight into the problem that they're solving. Andromeda provides a wide assortment of graphical analysis techniques to help the user distill and understand this information.

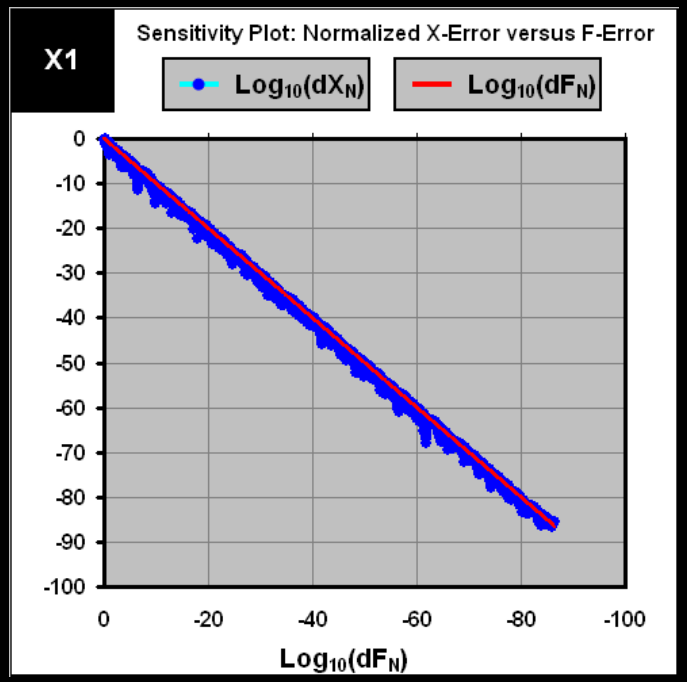


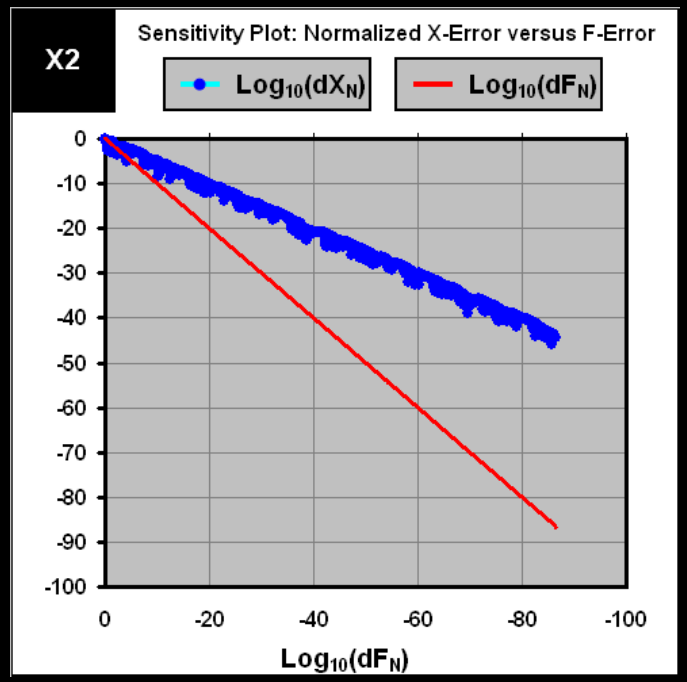
After the user selects all of the different types of plots that are of interest, clicking on the "Create Plot Summary" button will generate a new worksheet containing all of the plots requested.

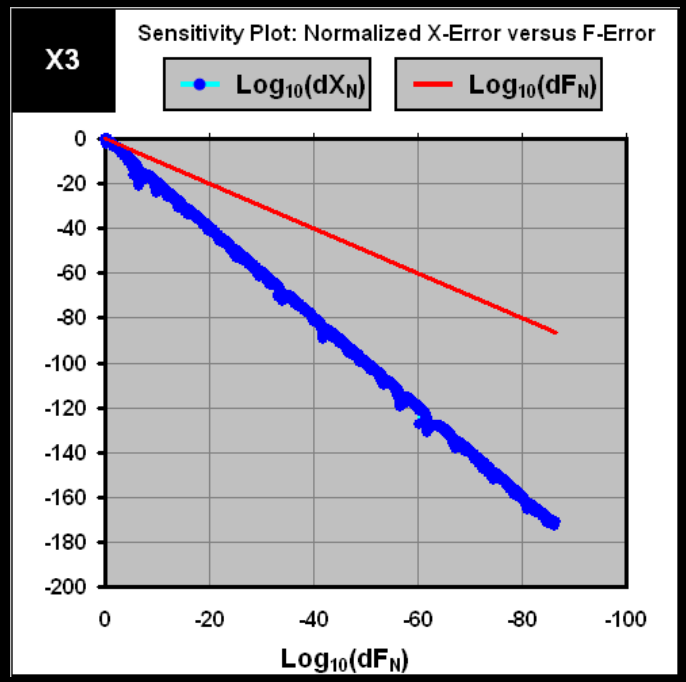
Graphical Analysis: Sensitivity Plots

Sensitivity plots make use of all the data generated during the course of particle convergence to determine the sensitivity of the function to each X variable as shown in the example below.

Solve: F1 = ABS(X1) + ABS(X2)^2 + ABS(X3)^0.5 = 0







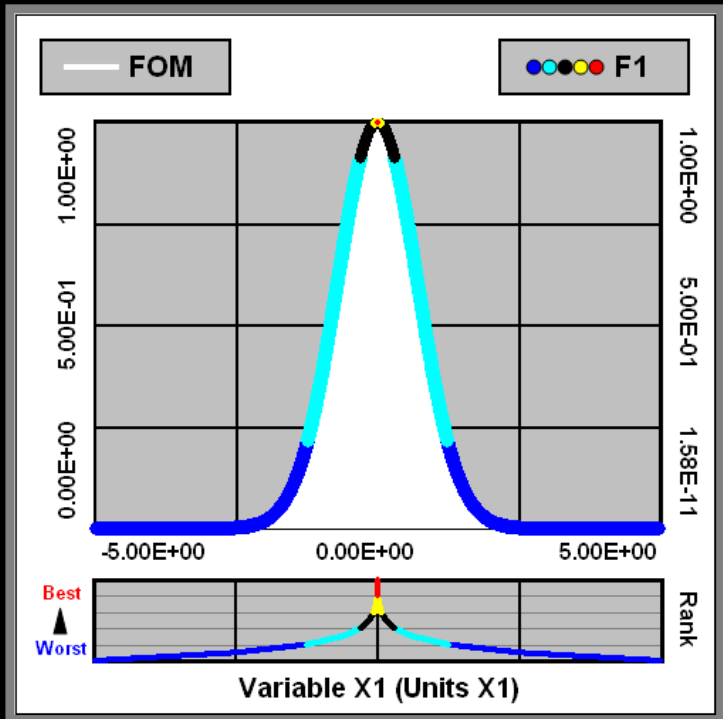
$F1 \propto |X1|$ $F1 \propto |X2|^2$ $F1 \propto |X3|^{1/2}$

The mathematical definitions of the Normalized Errors in X and F are presented in the Appendix.

Graphical Analysis: FOM and Function Plots

FOM = Figure Of Merit
= 1 at Best solution value
= 0 at Worst solution value
and is plotted in white.

Solve:
F1 = Exp(-(X1^2))
= Max

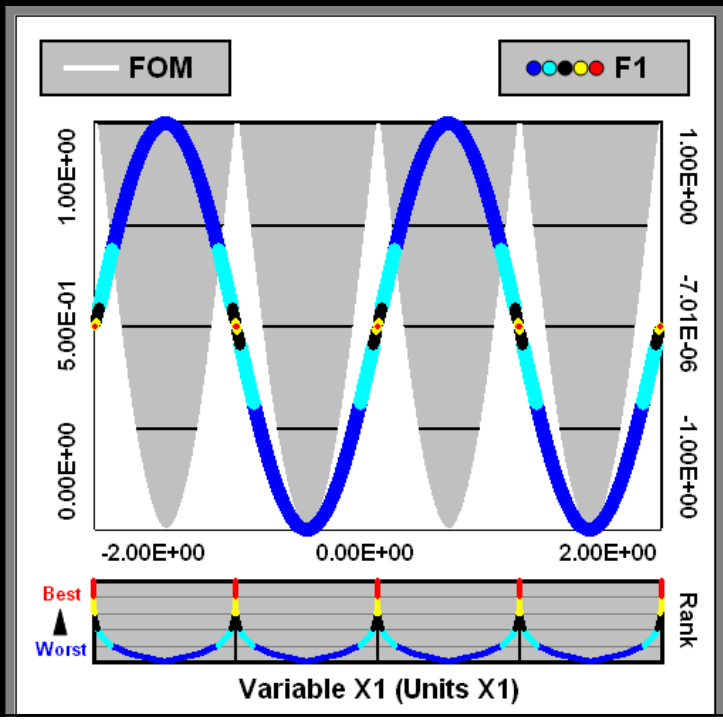


The rank order of the functional values of the particles is indicated by a color code which is used throughout the application:

- Best 20% of Particles
- 60-80% range
- Median 20% of Particles
- 20-40% range
- Worst 20% of Particles

The X values are rank ordered from **Best** to **Worst** in the marginal plot located just below the X-axis.

Solve:
F1 = Sin(πX1)
= 0



Each of the 5 color codes contains exactly the same number of particles (20% of the total), so the plots show that the particles have much higher densities in the solution areas where they are converging.

Graphical Analysis: Contour Plots

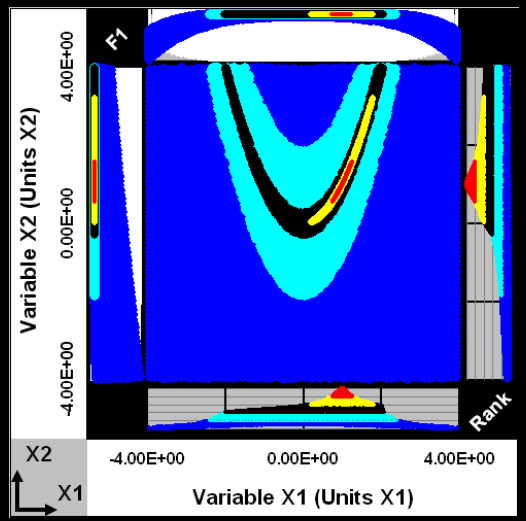
The large plot located at the center of each graph is a Contour plot showing the spatial coordinates of all the color coded particles (using the same color code defined on the previous slide).

The marginal plots along each side of the Contour plot are the corresponding Function and Rank plots for each X (also defined on the previous slide).

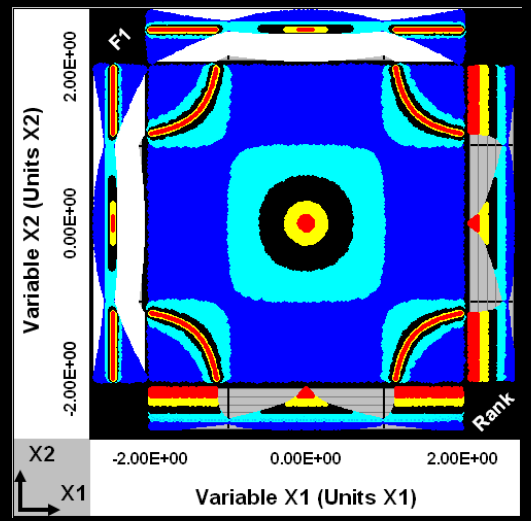
Since each color code contains exactly the same number of particles (20% of the total), the particle distributions exhibit much greater density in the solution areas where they are converging. Contour plotting the distribution of a large number of particles after a short number of runs is an excellent way to gain an understanding of complex functions in a multidimensional space!

(The name of the software – Andromeda – is derived from the fact that a contour plot of the particles converging on a solution once reminded me of a picture I had seen of that galaxy.)

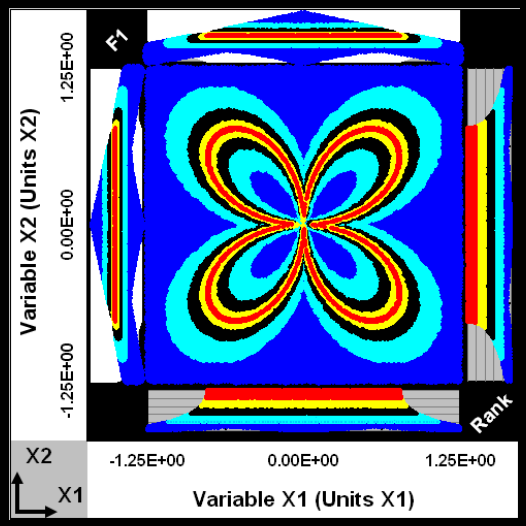
Solve: (Rosenbrock's Function)
F1 = 100(X2-X1^2)^2 + (1-X1)^2
= Min



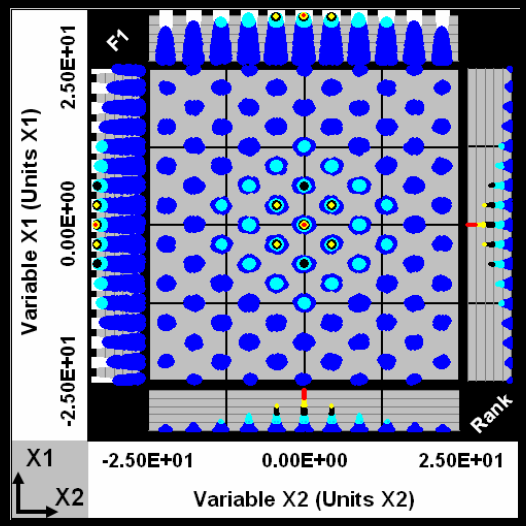
Solve:
F1 = Product(1-X1:X2^2)
= 1.0



Solve: (in Polar Coordinates)
F1 = R – Abs(Sin(2θ))
= 0.0



Solve:
F1 = Griewangk's Function (2D)
= Min



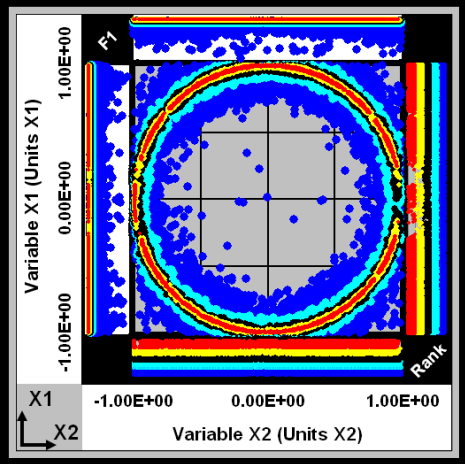
Graphical Analysis:

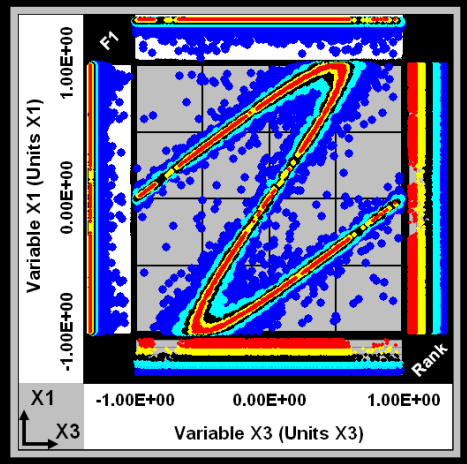
Matrix Contour Plots

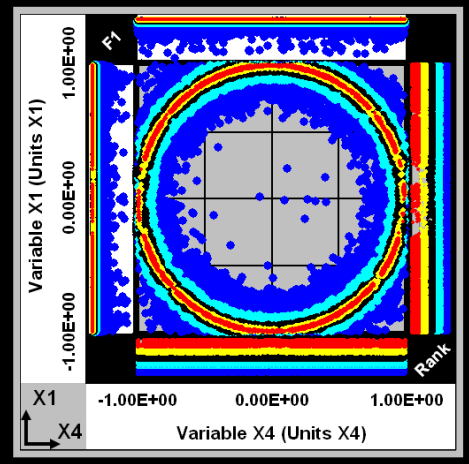
(X1,X2) (X1,X3) (X1,X4)

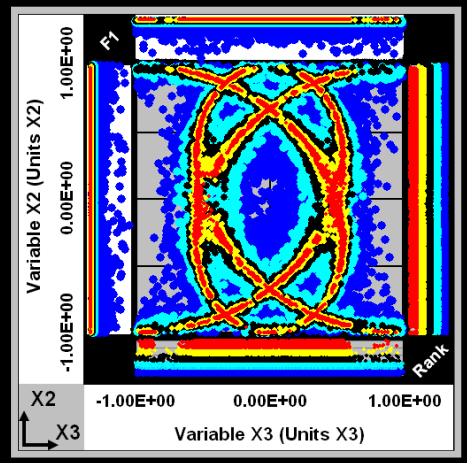
(X2,X3) (X2,X4)

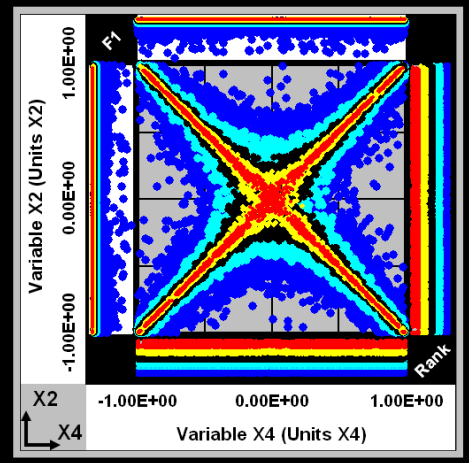
(X3,X4)

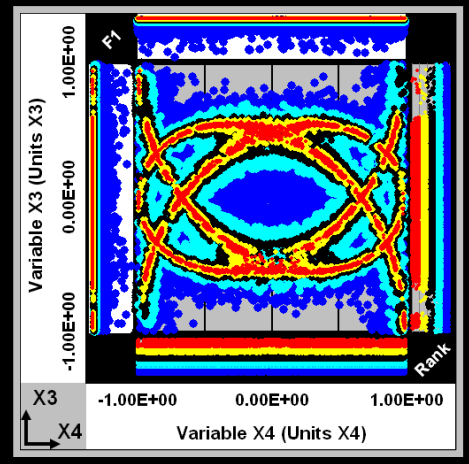












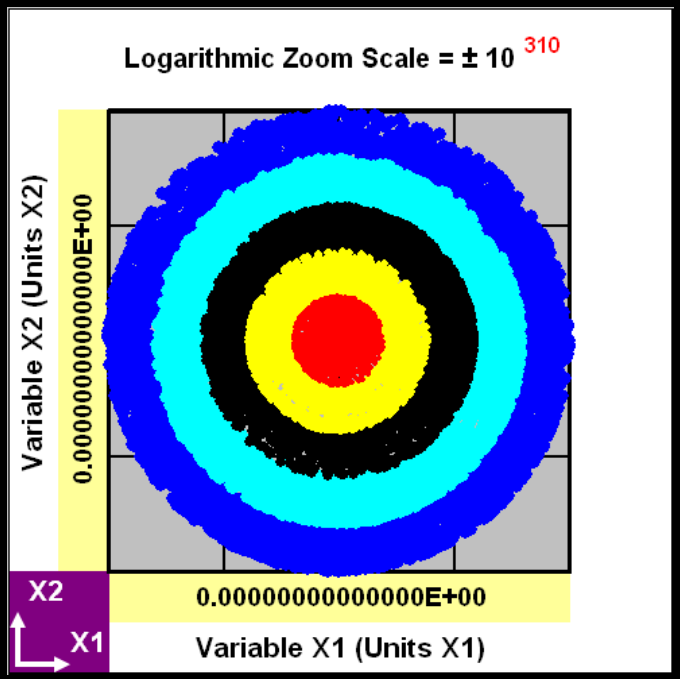
An example of matrix contour plotting the particle positions after several iterations of the PSE algorithm (10 runs in this case) to reveal the detailed structure of the solutions to the 4-Dimensional optimization problem shown below.

Solve: $F1(X1, X2, X3, X4) = |X1^2 + X2^2 - 1| + |X1 - \sin(\pi(X1-X3))| + |X2^2 - X4^2| = \text{Min}$

Graphical Analysis: Logarithmic Zoom Plots

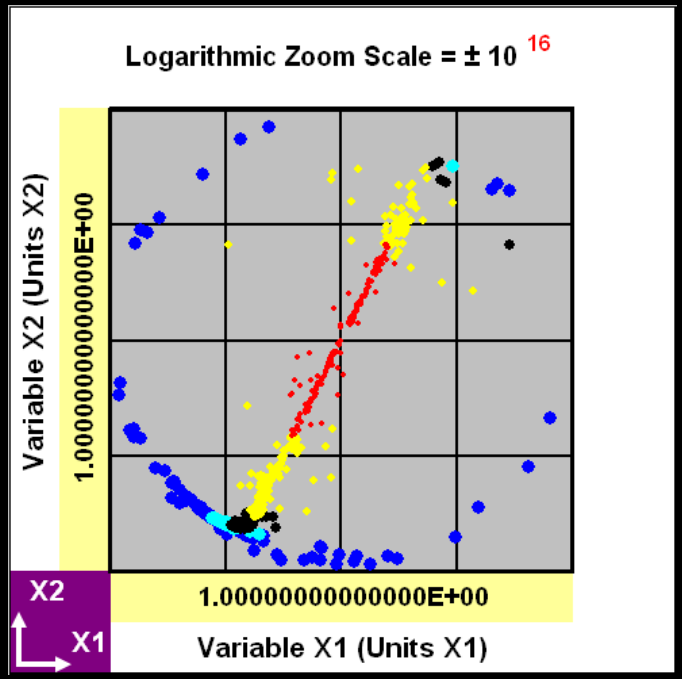
The Logarithmic Zoom plot in Andromeda makes use of a logarithmic transformation of the X variables which preserves angle to allow the user to follow the particle distribution all the way down to the best solution through many orders of magnitude in spatial scale. The numerical values highlighted in yellow on each axis are the X values of the best solution, and they define the center of each plot.

Solve:
F1 = Abs(X1) + Abs(X2) = 0.0



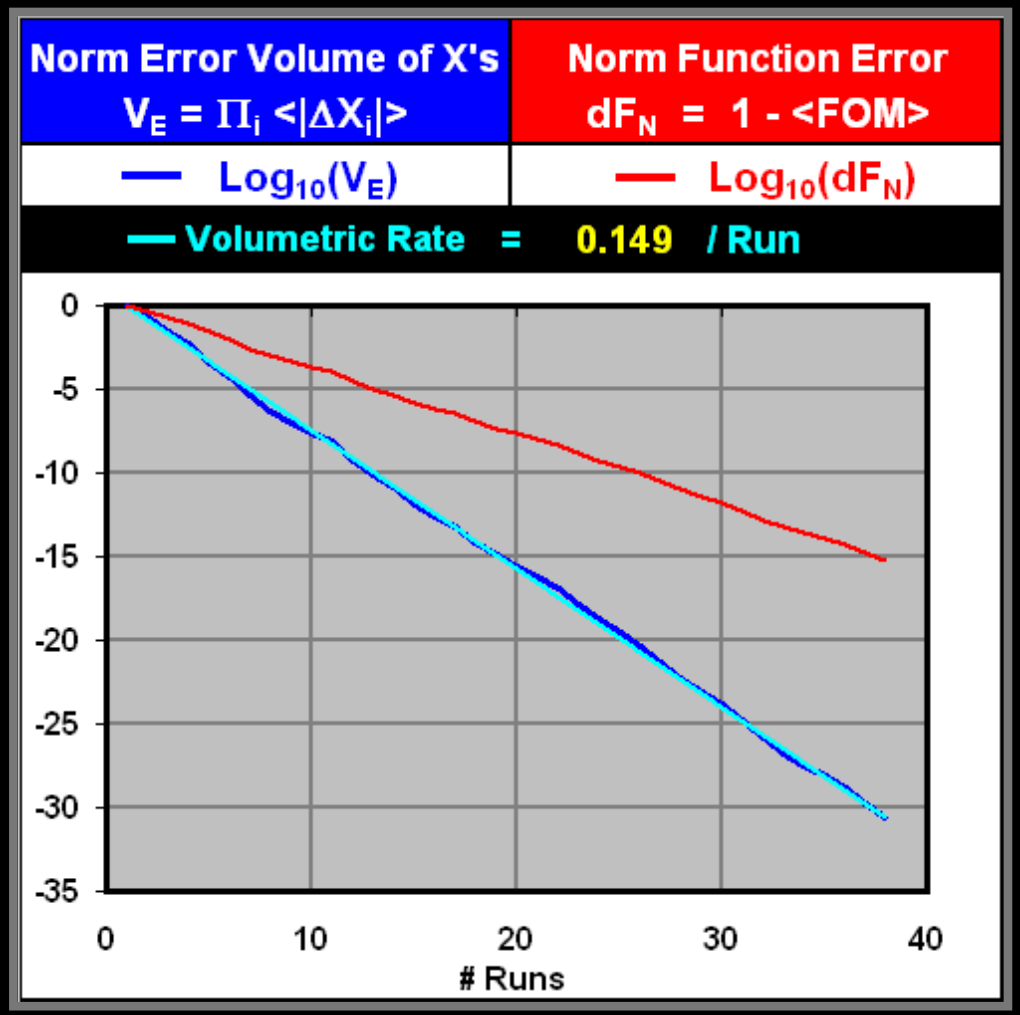
In this first example, the particles converge down through 310 orders of magnitude in spatial scale to the solution (representing the machine value of Zero in Excel).

Solve: (Rosenbrock's Function)
F1 = 100(X2-X1^2)^2 + (1-X1)^2 = 0.0



In this second example, the particles converge down through 16 orders of magnitude in spatial scale to the solution (having the 15 decimal place precision of Excel).

Graphical Analysis: The Convergence Plot



The convergence plot is provided as a tool for those who wish to study the convergence properties of the PSE algorithm. It shows the reduction in both the Error Volume of the X's and the Function Error versus the number of runs (iterations).

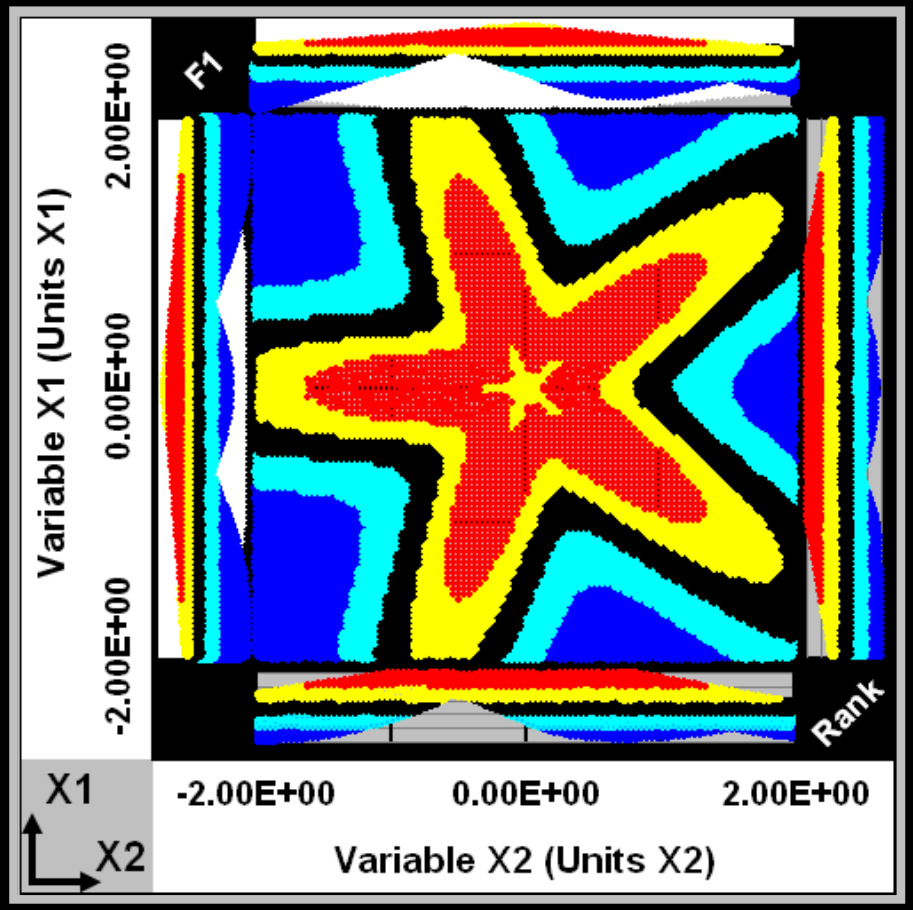
The "Volumetric Rate" shown on the plot header is the factor by which the Error Volume is reduced (on average) for each run.

The convergence properties of the PSE algorithm are reviewed in the Appendix.

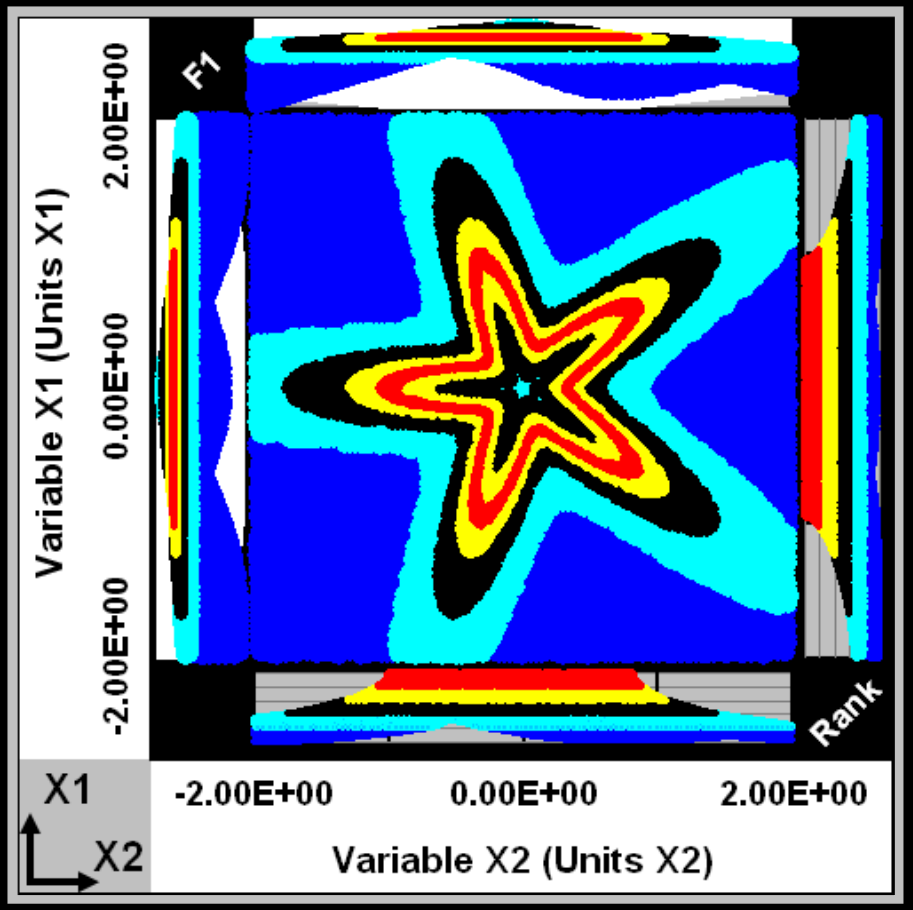
Using Andromeda as a "smart" plotting tool

As the particles converge with each iteration, their density greatly increases in the areas that are of specific interest, and thus the contour plots can also be used as a "smart" plotting tool to gain an understanding of complex functions in multidimensional spaces.

Solve: (in Polar Coordinates)
$F1 = R * [1 + 0.5 * \sin(5\theta)] = 0.5$



Runs = 1



Runs = 5

Single Function Optimization

In this section, we'll review the following topics and examples:

Excel Precision

Solve: F1(X1) = 1 – (X1^2)
= Maximum

Andromeda

F1=1-(X1^2)

Functions		Goal	Target	Wt	Best Value	Error
Function1 (Units F1)	F1	Max	1.00E+00	1	1.00000000000000E+00	0.E+00

Input Variables		Min	Max	B	Best Value	± Range
Variable1 (units X1)	X1	-1.00E+00	1.00E+00	Y	-6.17979678512755E-09	4.E-08

Termination: Specified Function Error Tolerance (Zero Error) Achieved

# Runs (Iterations) =	16
Particle Population (Np) =	2
Function Evals =	32
Hard Constraint Evals =	0

Execution Time Components (sec)	
Function Evals =	0.00
PSE Algorithm =	0.00
Total RunTime =	0.09

Search Mode
Rapid Convergence
Memory Used
12.94 MB

The exact solution, of course, is X1 = 0 and F1(X1) = 1, but we find that Excel reports F1(X1) = 1 when |X1| is on the order of 10^{-8} or smaller. The reason is that Excel has a precision of 15 digits, and therefore subtracting |X1|^2 = 10^{-16} from 1 appears to equal 1.

Excel Smallest Number

Solve: F1(X1) = X1
= 0

Andromeda

F1=X1

Functions		Goal	Target	Wt	Best Value	Error
Function1 (Units F1)	F1	F1 =	0.00E+00	1	0.00000000000000E+00	0.E+00

Input Variables		Min	Max	B	Best Value	± Range
Variable1 (units X1)	X1	-1.00E+00	1.00E+00	Y	0.00000000000000E+00	0.E-01

Termination: Specified Function Error Tolerance (Zero Error) Achieved

# Runs (Iterations) =	620
Particle Population (Np) =	2
Function Evals =	1,240
Hard Constraint Evals =	0

Execution Time Components (sec)	
Function Evals =	0.19
PSE Algorithm =	0.03
Total RunTime =	0.50

Search Mode
Rapid Convergence
Memory Used
12.98 MB

In this example, we are not limited by the 15 digit precision of Excel and appear to have obtained an "exact" solution of X1 = 0. But what has actually happened is that the particles have tunneled down through more than 300 orders of magnitude in X1 to reach the smallest number that Excel can handle (which is on the order of 10^{-308}).

Excel Largest Number

Solve: F1(X1) = X1
= Maximum

Andromeda

F1=X1

Functions		Goal	Target	Wt	Best Value	Error
Function1 (Units F1)	F1	Max		1	1.27398975689877E+298	

Input Variables		Min	Max	B	Best Value	± Range
Variable1 (units X1)	X1	-3.81E-01	1.27E+298	N	1.27398975689877E+298	0.E+00

Termination: Specified Average X Range Tolerance (Zero) Achieved

# Runs (Iterations) =	90
Particle Population (Np) =	2
Function Evals =	180
Hard Constraint Evals =	0

Execution Time Components (sec)	
Function Evals =	0.00
PSE Algorithm =	0.00
Total RunTime =	0.13

Search Mode
Rapid Convergence
Memory Used
13.02 MB

In this example, we run into another limit of Excel, namely:
the largest number that Excel can handle
(which is on the order of 10^{+308}).

Example: Minimizing Griewangk's Function

Griewangk's Function is commonly used to test global optimization routines because it contains a global minimum and many widespread local minima.

For n dimensions, Griewangk's function is defined as:

$$F(X_1, X_2, ..., Xn) = 1 + \sum_{i=1}^{n} \frac{X_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{X_i}{\sqrt{i}}\right)$$

and the global minimum is F = 0 at the origin ($X_i = 0$).

We will solve this problem in n = 1, 2, and 3 dimensions, and we will also examine the results when certain constraints are imposed on the solution.

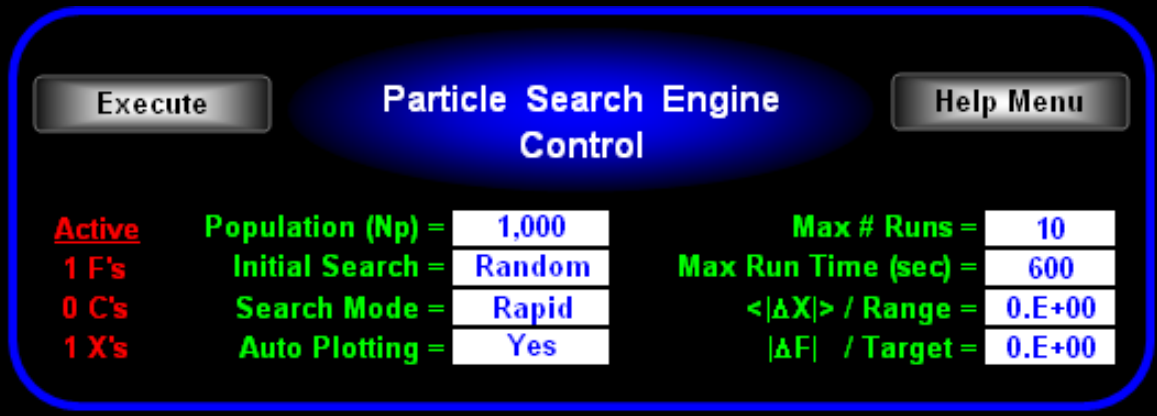
Example: Minimizing Griewangk's Function (1-D)

We wish to minimize the function

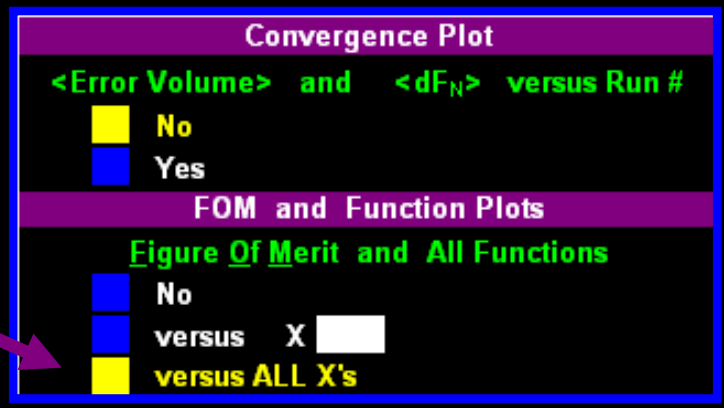
$$F1(X1) = 1 + (X1^2)/4000 - \cos(X1)$$

on the interval $|X1| \leq 15$.

Initially, we're not sure how many solutions to expect, so we'll use a relatively large number of particles (1000) for just a small number of iterations (10) to scope out the nature of the problem we're solving.

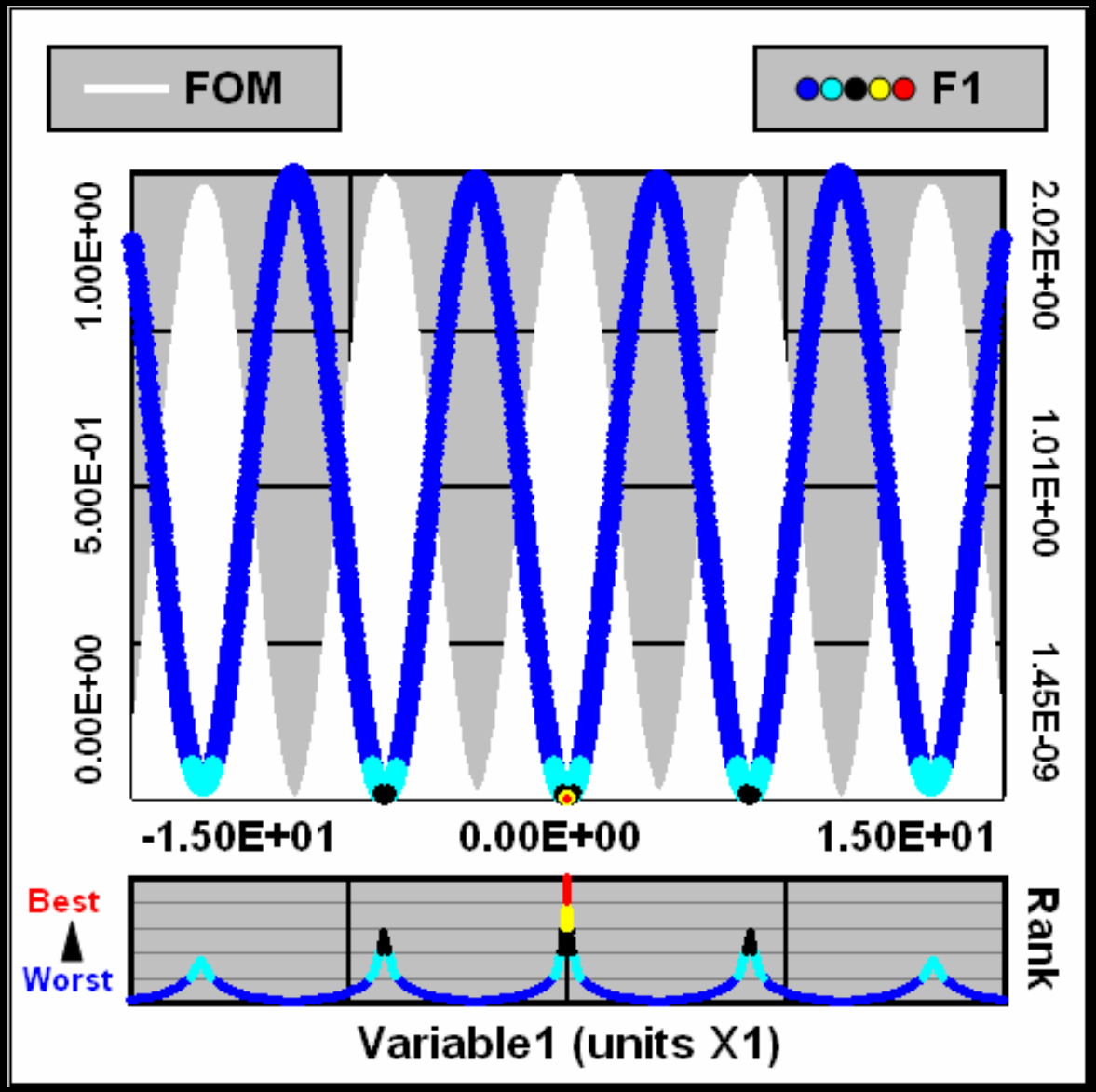


Also, on the Graphics worksheet, we'll request a FOM and Function plot to see how many local optima we're dealing with in this problem.



Example: Minimizing Griewangk's Function (1-D)

We then find that there is a global minimum at the origin, and 4 local minima within the region |X1| ≤ 15.



Example: Minimizing Griewangk's Function (1-D)

Andromeda

F1=1+(X1^2)/4000-COS(X1)

Functions		Goal	Target	Wt	Best Value	Error
Function1 (Units 1)	F1	Min	0.00E+00	1	0.00000000000000E+00	0.E+00

Input Variables		Min	Max	B	Best Value	± Range
Variable X1 (Units X1)	X1	-1.00E+00	1.00E+00	Y	2.944109084447O2E-08	4.E-08

Termination: Specified Function Error Tolerance (Zero Error) Achieved

# Runs (Iterations) =	16
Particle Population (Np) =	2
Function Evals =	32
Hard Constraint Evals =	0

Execution Time Components (sec)	
Function Evals =	0.02
PSE Algorithm =	0.00
Total RunTime =	0.17

Search Mode
Rapid Convergence
Memory Used
7.02 MB

The global solution at the origin can then be obtained by allowing those 1000 particles to fully converge (which takes about 16 runs in this case),
OR (much more efficiently)
reduce the search area to exclude the other local minima (say $|X1| \leq 1$) and use the minimum number of particles (Np = Pod Size = 2) to find the solution in only 32 function evaluations, as shown above.

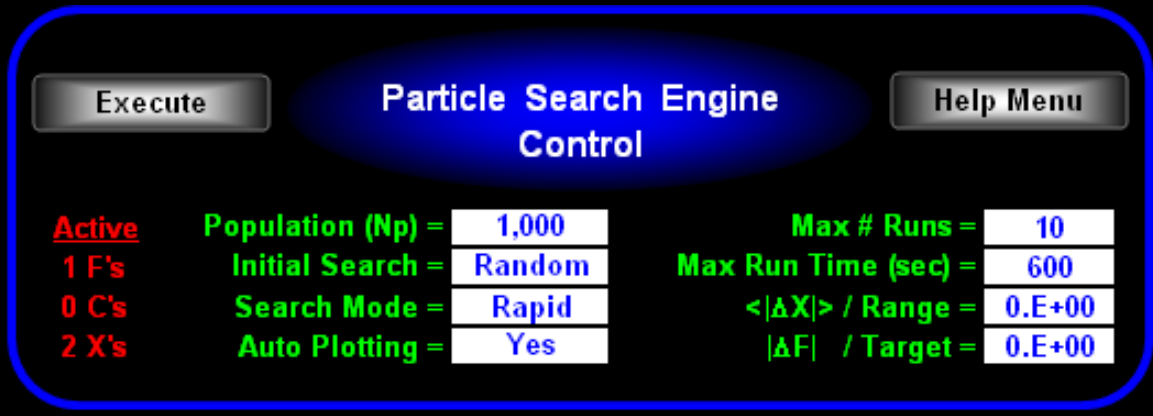
Example: Minimizing Griewangk's Function (2-D)

We wish to minimize the function

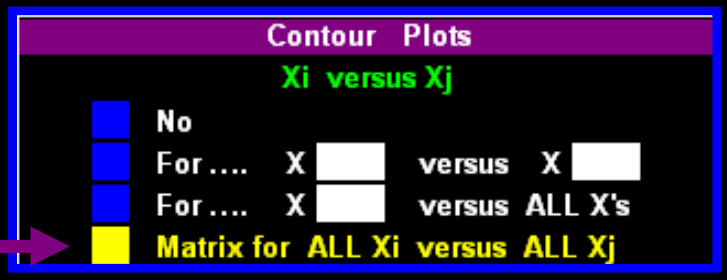
$$F1(X1,X2) = 1 + (X1^2+X2^2)/4000 - \cos(X1)\cos(X2/\sqrt{2})$$

over the area $|X_i| \leq 15$.

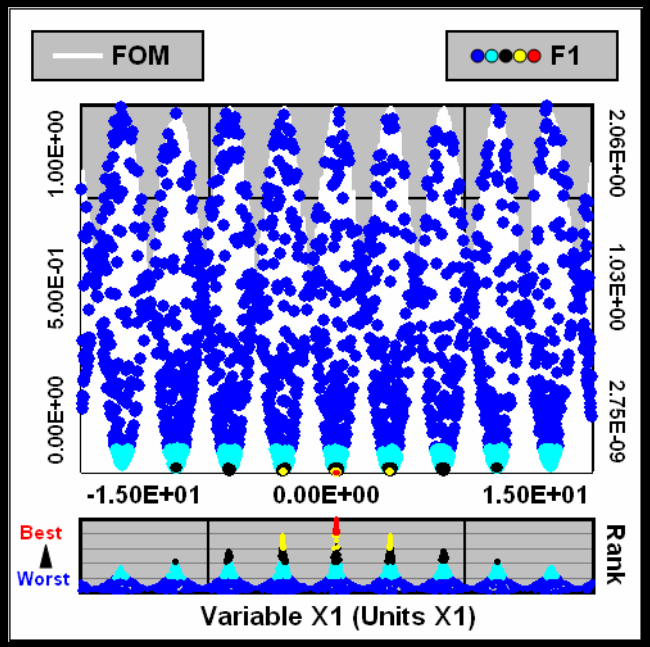
Once again, we're not sure how many solutions to expect, so we'll use a relatively large number of particles (1000) for just a small number of iterations (10) to scope out the nature of the problem we're solving.

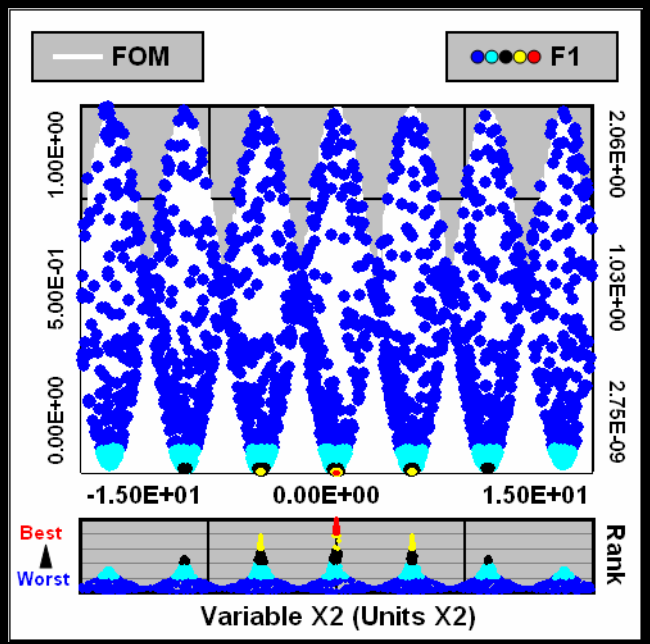


On the Graphics worksheet, we'll request a Contour plot in addition to the FOM and Function plots to see how many local optima we're dealing with in this problem.

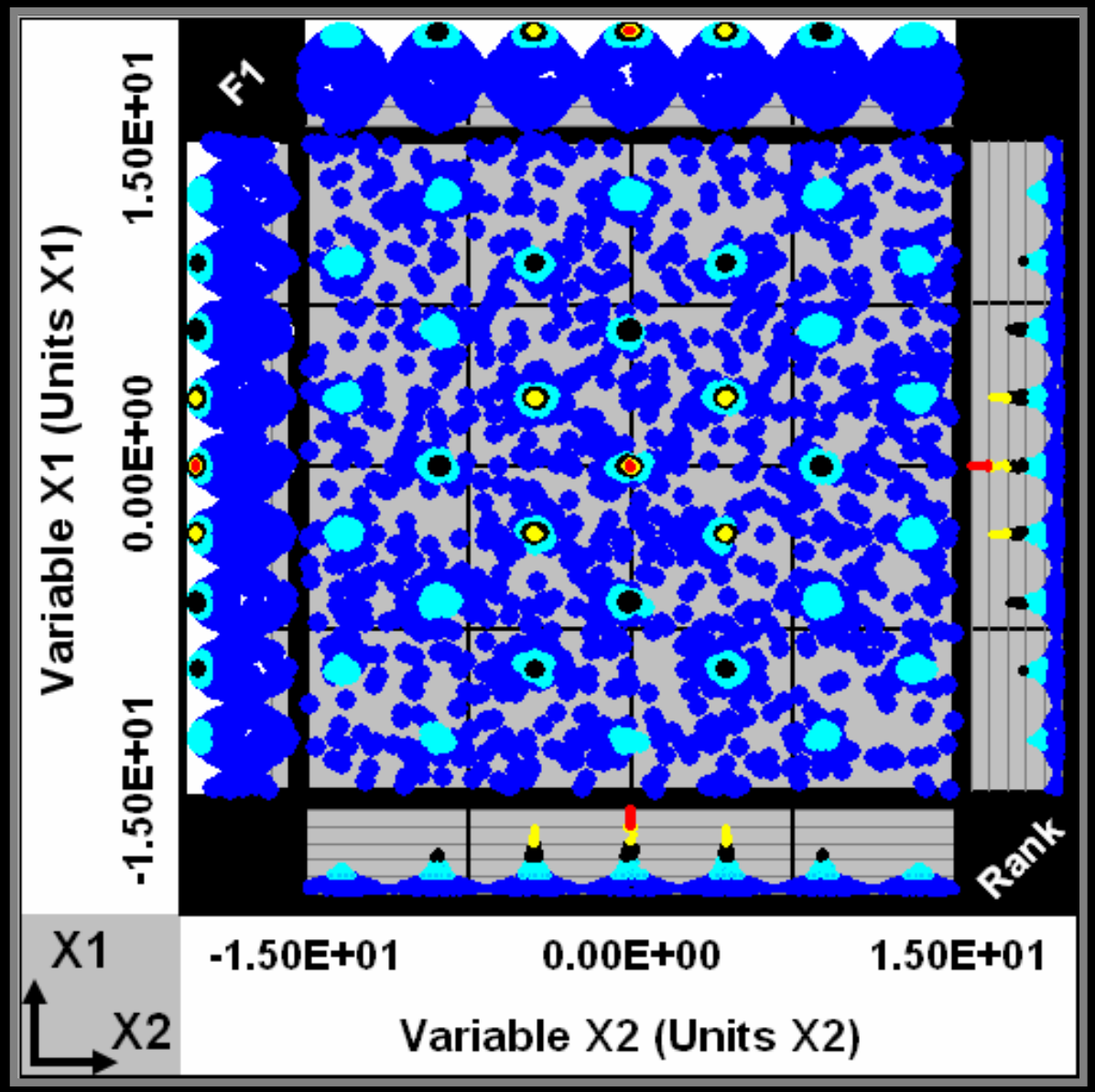


Example: Minimizing Griewangk's Function (2-D)



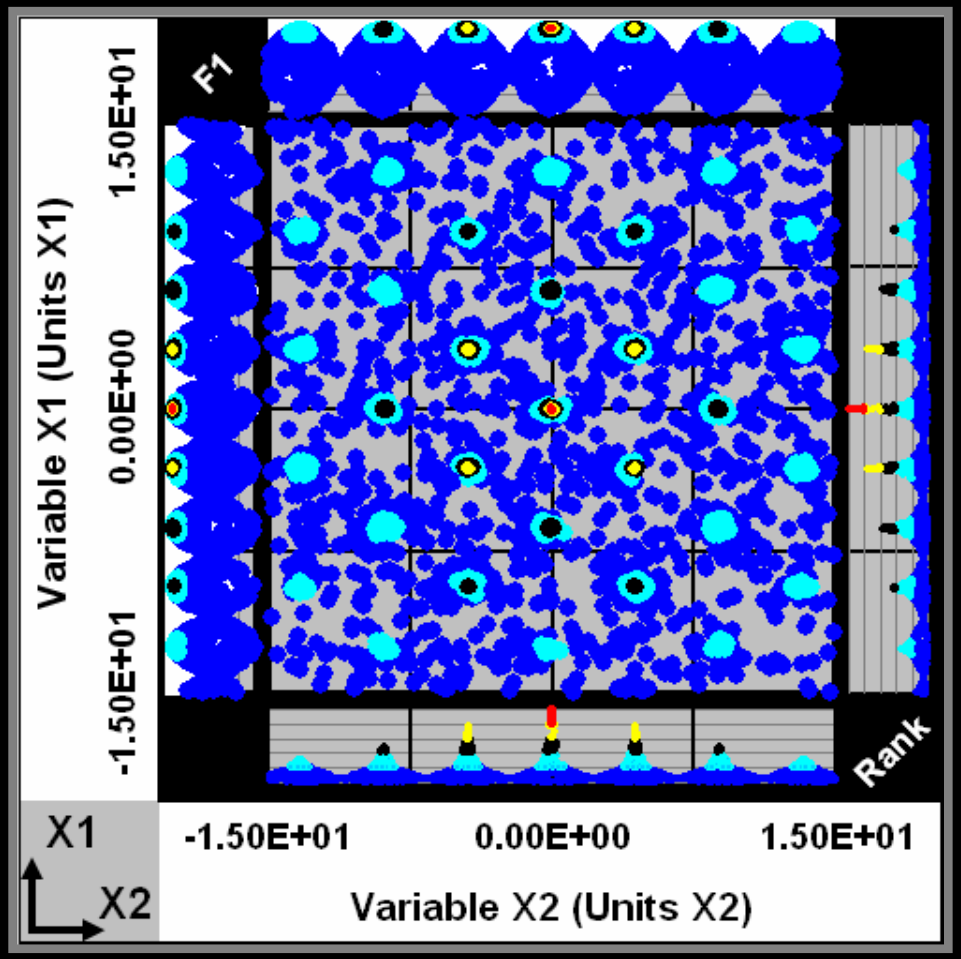


We then find that there is a global minimum at the origin, and 30 local minima within the area $|Xi| \leq 15$.

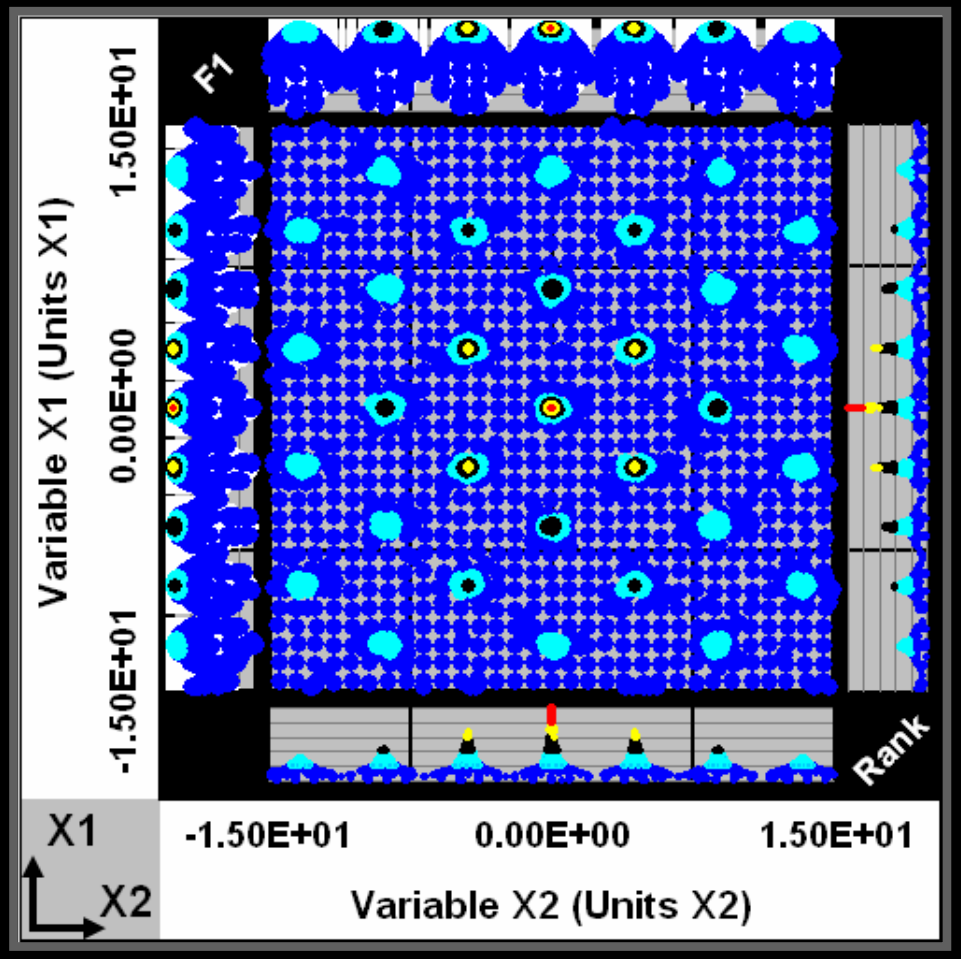


Example: Minimizing Griewangk's Function (2-D)

The "Array" option for the initial search was described in Chapter II, and we'll demonstrate its use in this 2-D example. As discussed earlier, given an initial population of Np = 1000 particles, the algorithm will assign $(31)^2 = 961$ of those particles to a spatial array which uniformly spans the 2-D space, and the remaining 39 particles will be distributed randomly.



Initial Search = "Random"



Initial Search = "Array"

Example: Minimizing Griewangk's Function (2-D, with constraints)

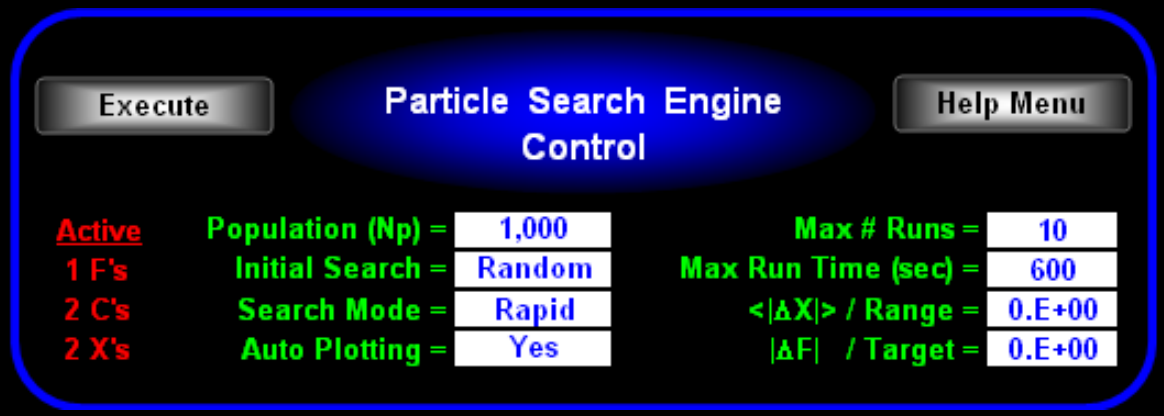
Now let's make the problem a little more interesting. We'll solve for the global minima of Griewangk's Function in 2-D within the annular region:

$$4 < R < 8$$

where $R = SQRT(X1^2+X2^2)$ is the radius.

Constraints

C's	Enter Equation	Select Constraint GOAL	Select Constraint Type	Wt.
C1	-8.00E+00	< 0	Hard	1
C2	-4.00E+00	> 0	Hard	1
C3				1
C4				1
C5				1
C6				1
C7				1



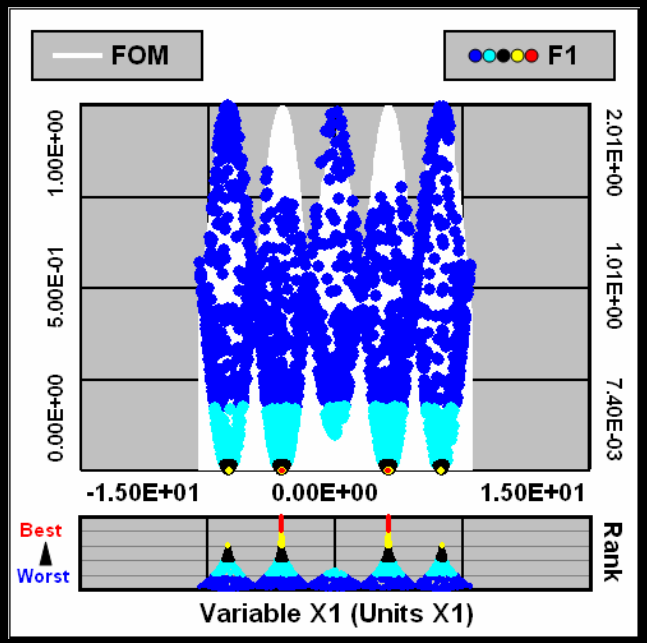
Select " Hard" as the Constraint Type for both C1 and C2 (meaning that C1 and C2 must be exactly satisfied, within the precision limits of Excel), and enter the following:

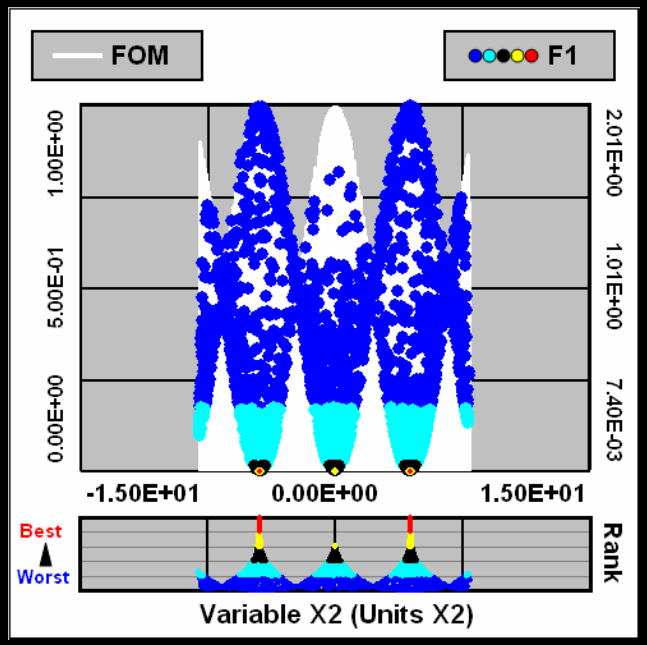
[C1: R < 8] C1 Equation: "=SQRT(X1^2+X2^2)-8" C1 Goal: " < 0 "

[C2 : R > 4] C2 Equation: "=SQRT(X1^2+X2^2)-4" C1 Goal: " > 0 "

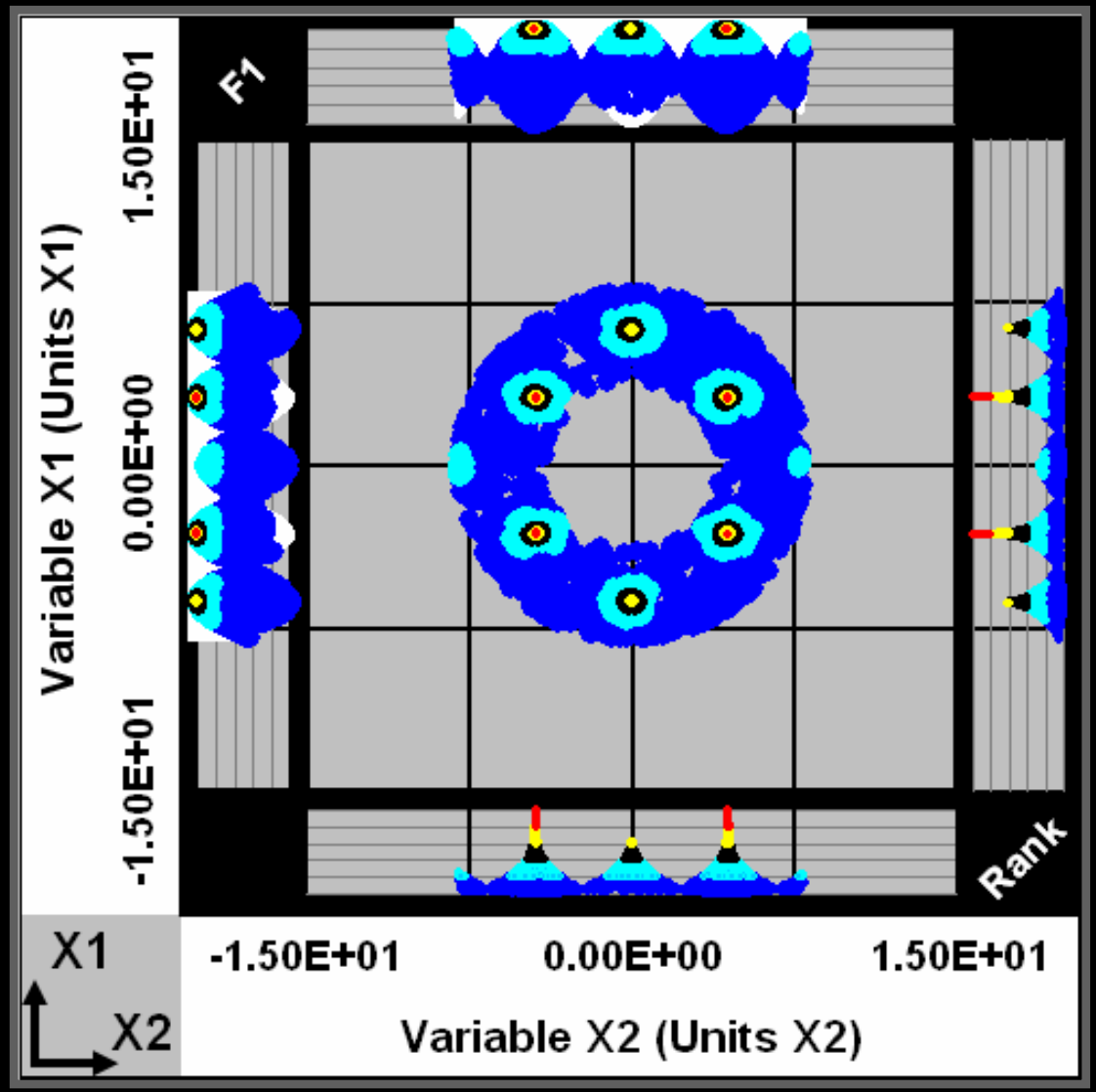
Then click the "Execute" button to run the program.

Example: Minimizing Griewangk's Function (2-D, with constraints)





We then find that there are 4 global minima and an additional 4 local minima within the annular region $4 < R < 8$



Example: Minimizing Griewangk's Function (3-D)

We wish to minimize the function

$$F1(X1,X2,X3) = 1 + (X1^2+X2^2+X3^2)/4000 - \cos(X1)\cos(X2/\sqrt{2})\cos(X3/\sqrt{3})$$

within the volume $|Xi| \leq 15$.

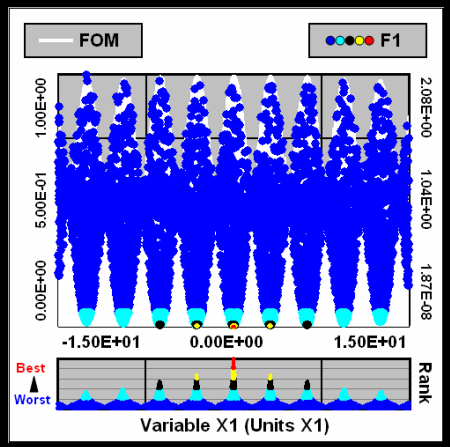
In this case, it turns out that a particle population Np = 1000 is actually not quite sufficient to check out all of the local optima. As we'll see on the next slide, there are roughly 300 local optima within the volume $|Xi| \leq 15$. In Chapter II of this user guide, we noted that the minimum number of particles that should be used to explore an optimization problem should be at least

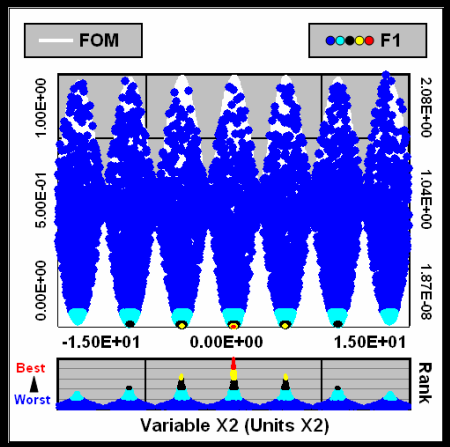
$$Np \geq (\text{Pod Size}) \times (\#\ \text{Local Optima})$$

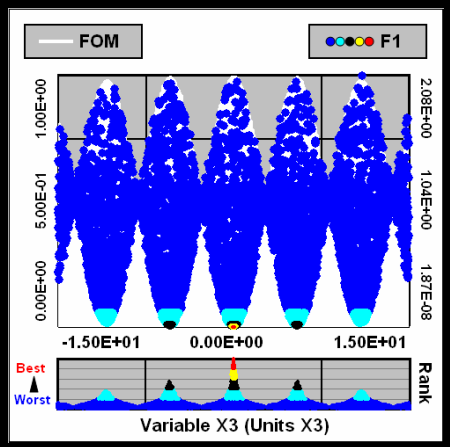
As shown previously on slide <24>, for a 3 dimensional problem the Pod Size is 10, and we therefore need a population size of at least Np = 10x300 = 3000 particles. On the next slide we'll show the results we obtain with Np = 3500 particles.

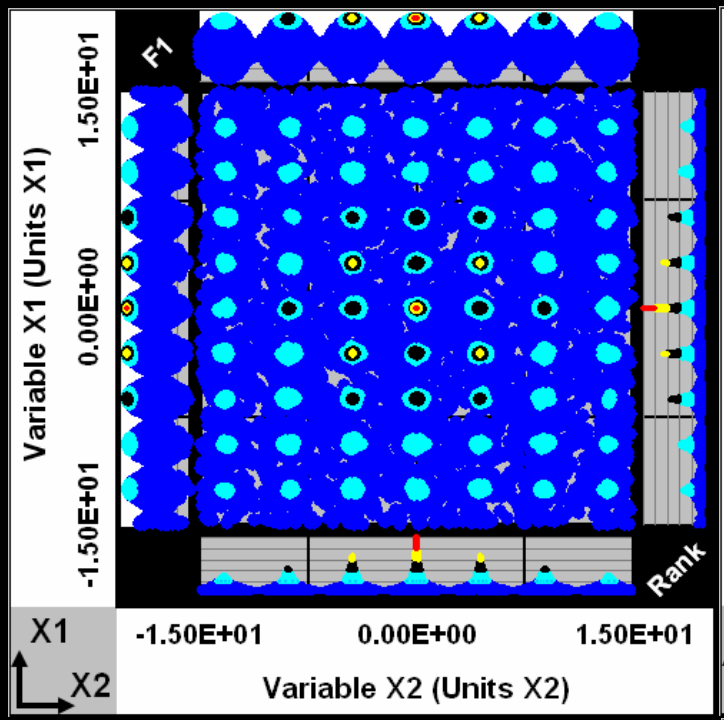
(In practice, one generally doesn't know how many local optima to expect in an optimization problem, and must therefore use trial and error to find out how many particles are sufficient to explore all of the local optima.)

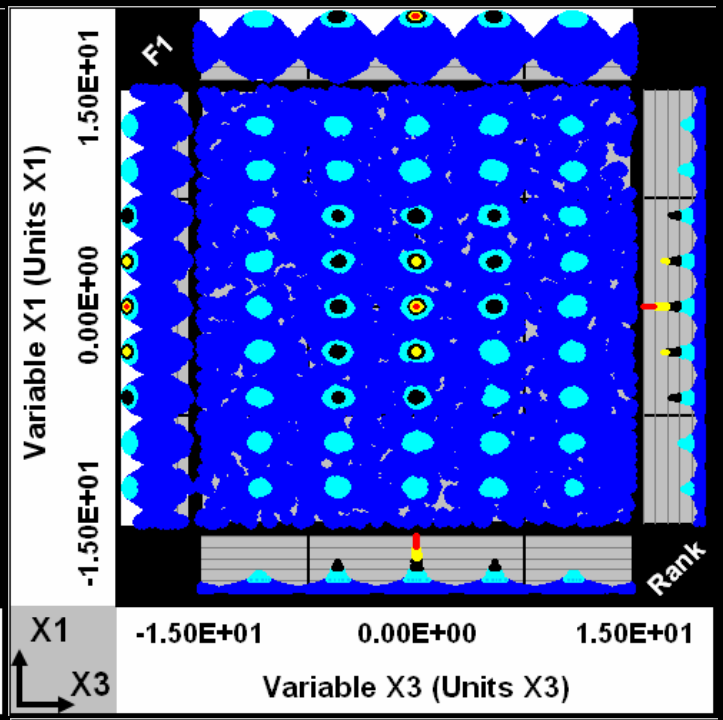
Example: Minimizing Griewangk's Function (3-D)





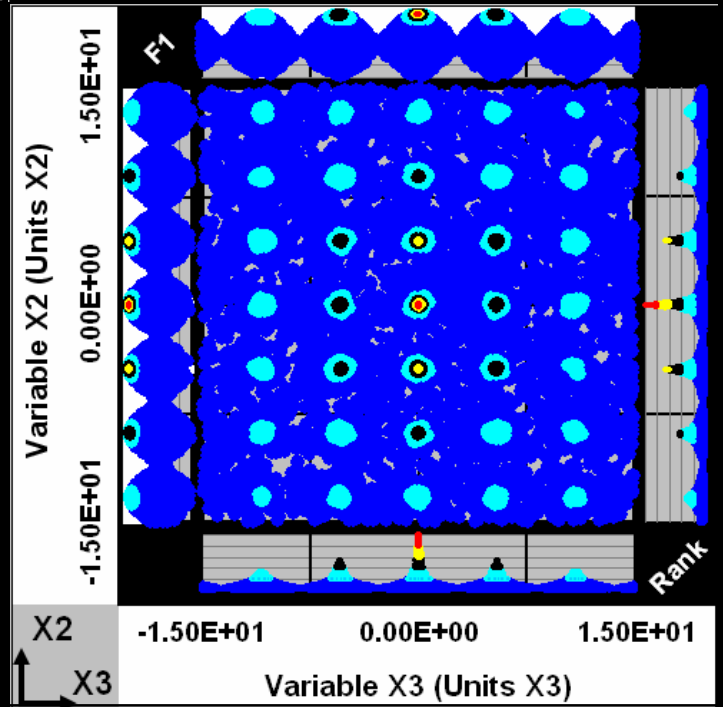




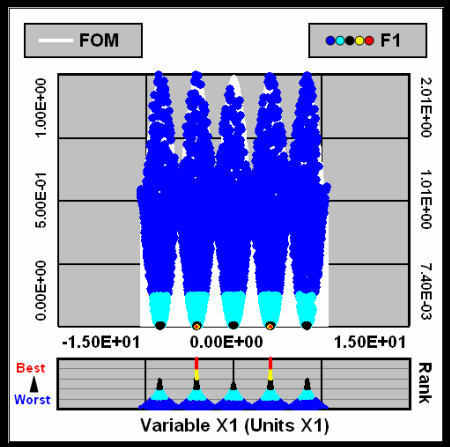


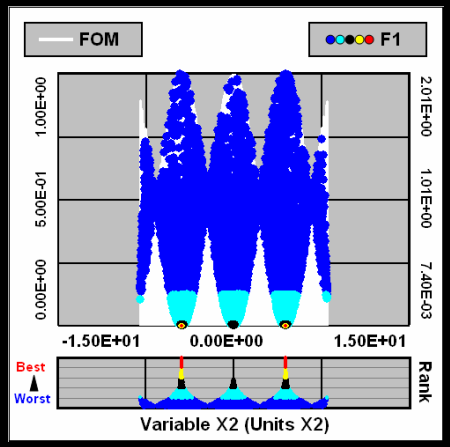
Here's the matrix of Contour plots obtained with Np = 3500 particles and 15 iterations of the algorithm.

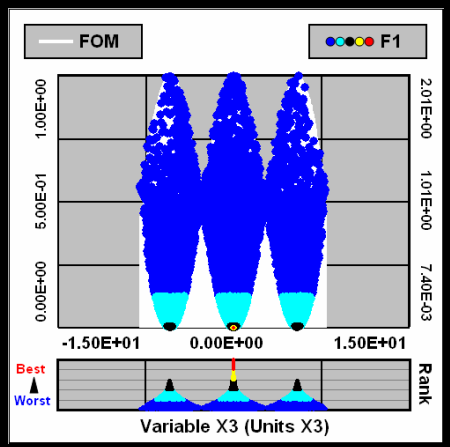
It reveals the global minimum at the origin, and over 300 local minima within the volume $|Xi| \leq 15$.

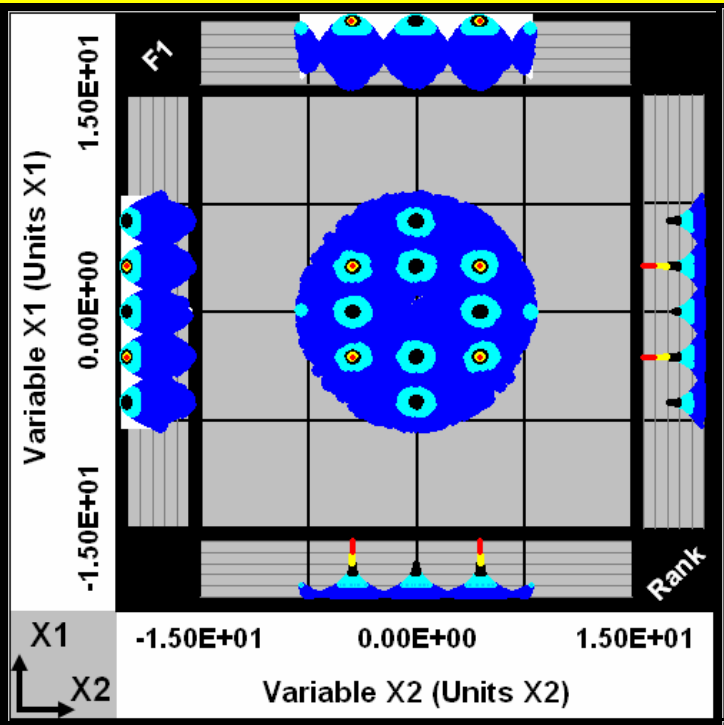


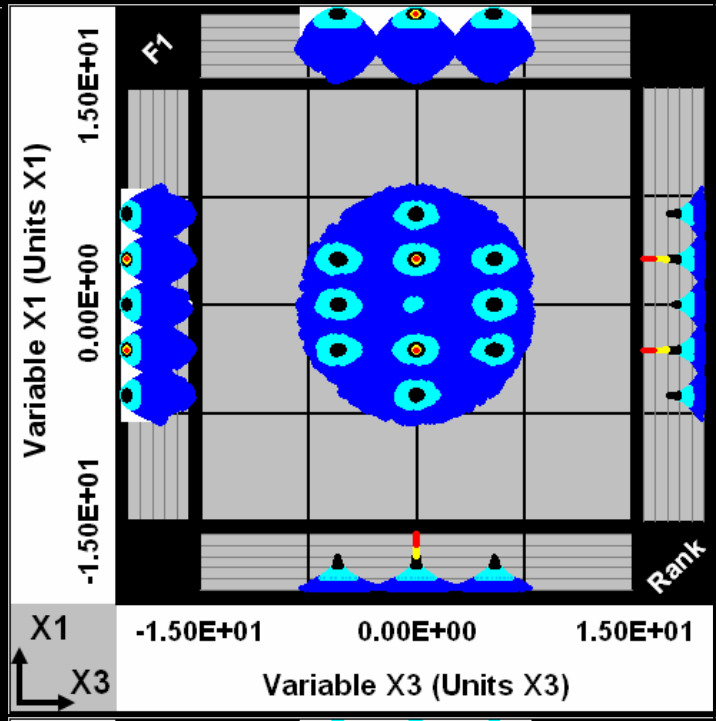
Example: Minimizing Griewangk's Function (3-D, with constraints)

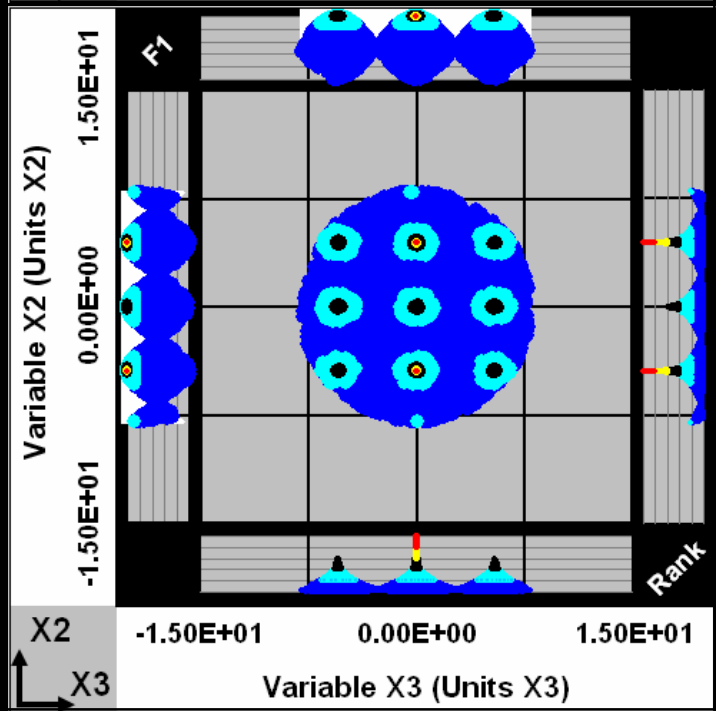












If we constrain the problem to the radial region

$4 < R < 8$

where

$R = \sqrt{X1^2+X2^2+X3^2}$

we find the same 4 global minima as in the previous 2-D constrained problem (X1 and X2 each having two solutions, and X3 at the origin)

Example: Minimizing Rosenbrock's Function

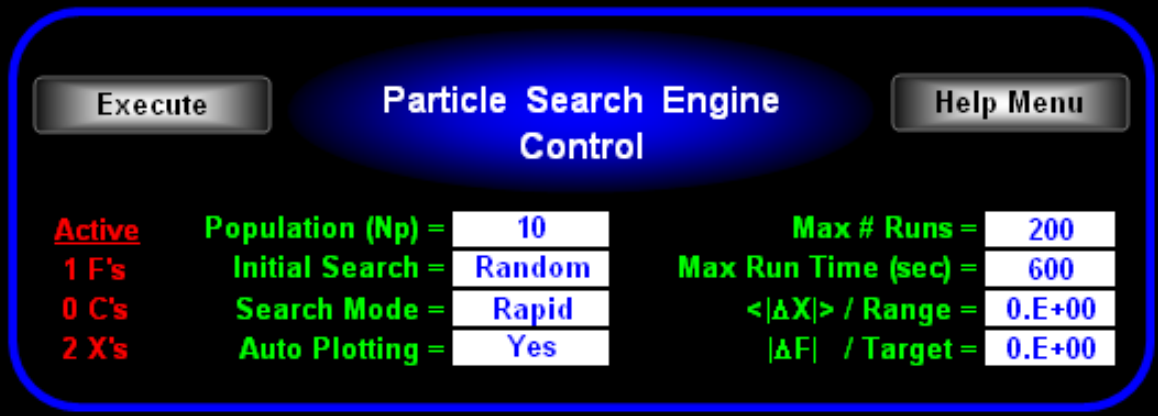
Rosenbrock's function is another function that is commonly used to test optimization routines, and is defined as:

$$F(X_1, X_2) = 100(X_2 - X_1^2)^2 + (1 - X_1)^2$$

It has a minimum value of F = 0 at X1 = X2 = 1.

Also known as the "banana" function, it can be quite a challenge for gradient methods because of the long narrow valley along the parabola X2=X1^2 which slowly winds down to the solution.

We'll solve this problem using the smallest possible particle population Np = 10 (the Pod size for 2-D problems).



Example: Minimizing Rosenbrock's Function

The exact solution (for 15 digits of precision) is found in 80 iterations

Andromeda

F1=100*(X2-X1^2)^2+(1-X1)^2

Functions		Goal	Target	Wt	Best Value	Error
Rosenbrock's Function	F1	Min	0.00E+00	1	0.000000000000000E+00	0.E+00

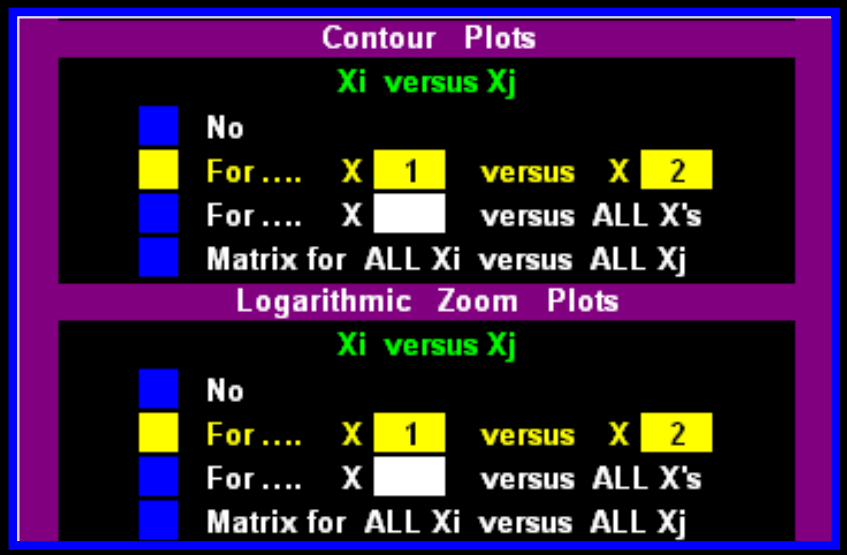
Input Variables		Min	Max	B	Best Value	± Range
Variable1 (Units X1)	X1	-5.00E+00	5.00E+00	Y	1.000000000000000E+00	3.E-14
Variable2 (Units X2)	X2	-5.00E+00	5.00E+00	Y	1.000000000000000E+00	7.E-14

Termination: Specified Function Error Tolerance (Zero Error) Achieved

# Runs (Iterations) =	80
Particle Population (Np) =	10
Function Evals =	800
Hard Constraint Evals =	0

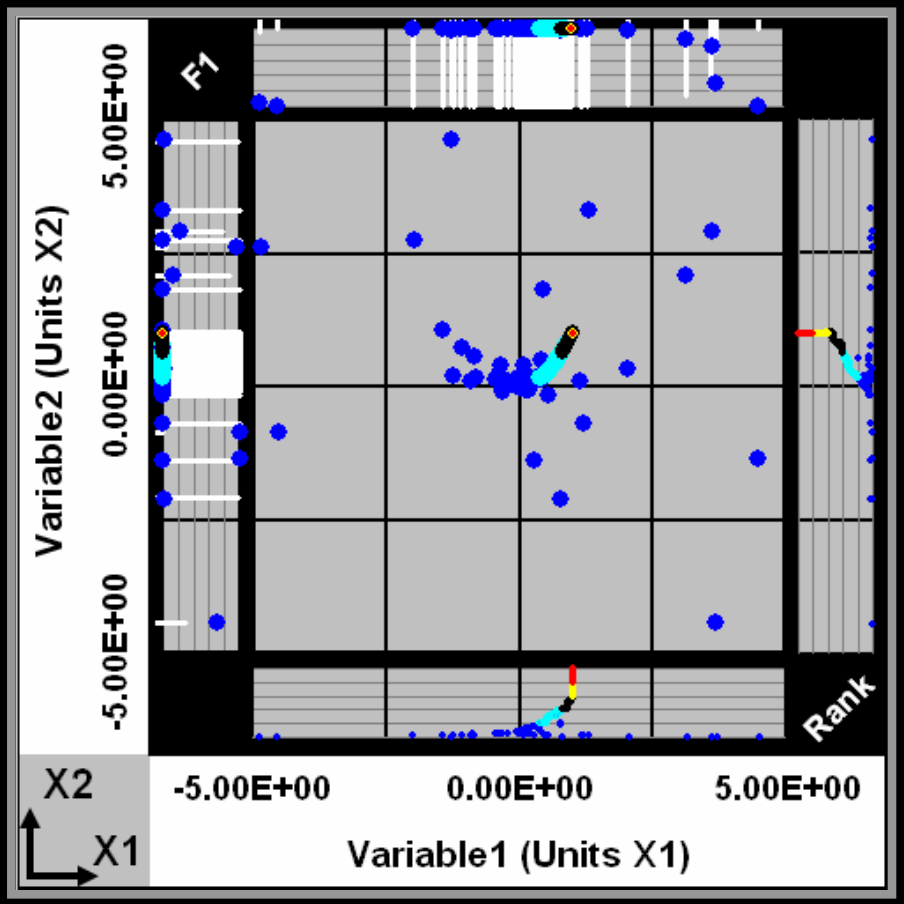
Execution Time Components (sec)	
Function Evals =	0.20
PSE Algorithm =	0.13
Total RunTime =	0.55

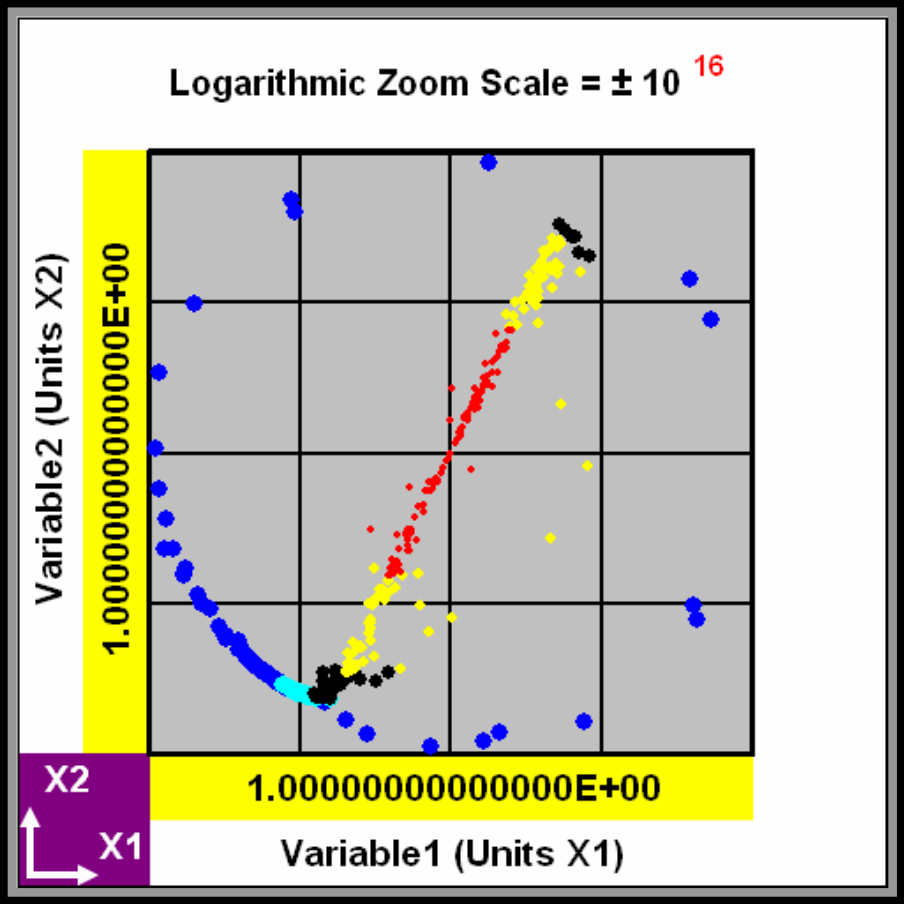
Search Mode
Rapid Convergence
Memory Used
16.73 MB



Let's now look at the data using a Contour plot and a Zoom plot.

Example: Minimizing Rosenbrock's Function





Contour

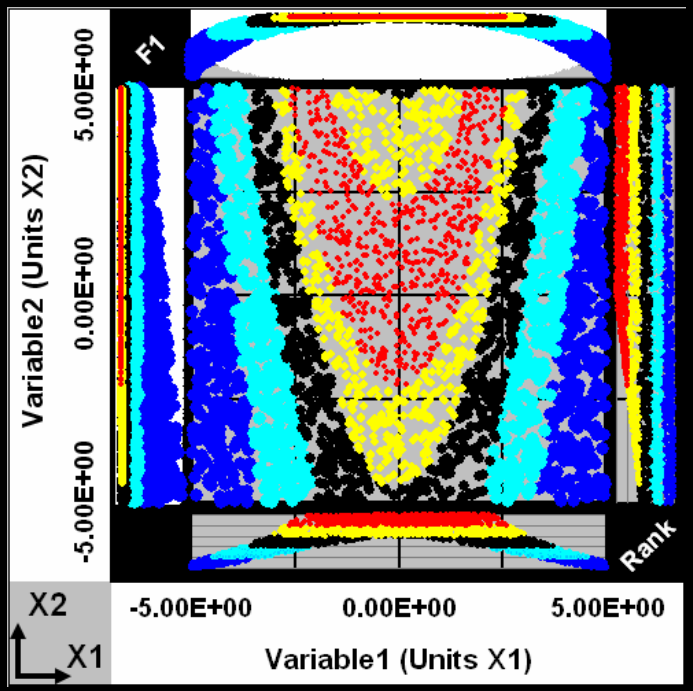
There's not very much to see on a linear spatial scale when using the minimum number of particles since most of the action takes place in the immediate area of the solution.

Zoom

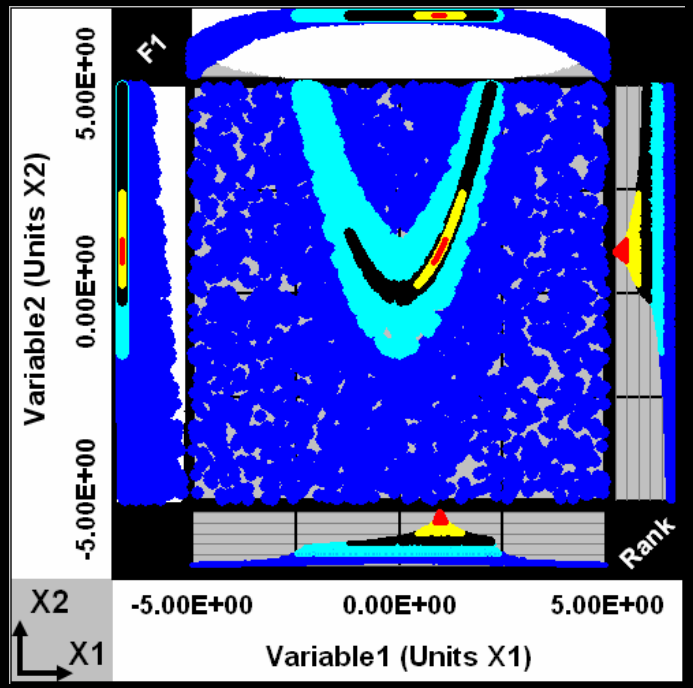
The Logarithmic Zoom plot shows the particles converging down to the solution (the numerical values highlighted in yellow) through 16 orders of magnitude in spatial scale.

Example: Minimizing Rosenbrock's Function

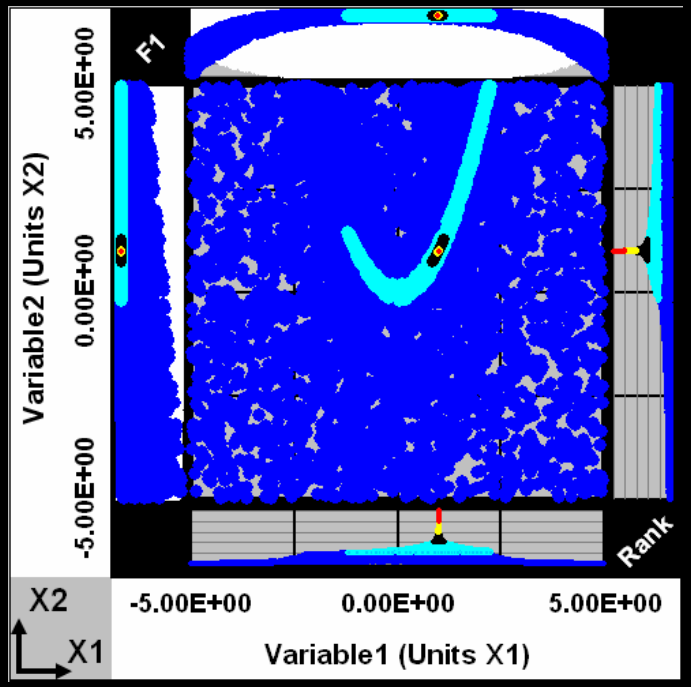
In order to get more detail on a linear spatial scale, just use a large particle population size and a small number of runs.



 # Runs = 1



 # Runs = 10



 # Runs = 20

Optimizing Discontinuous Functions

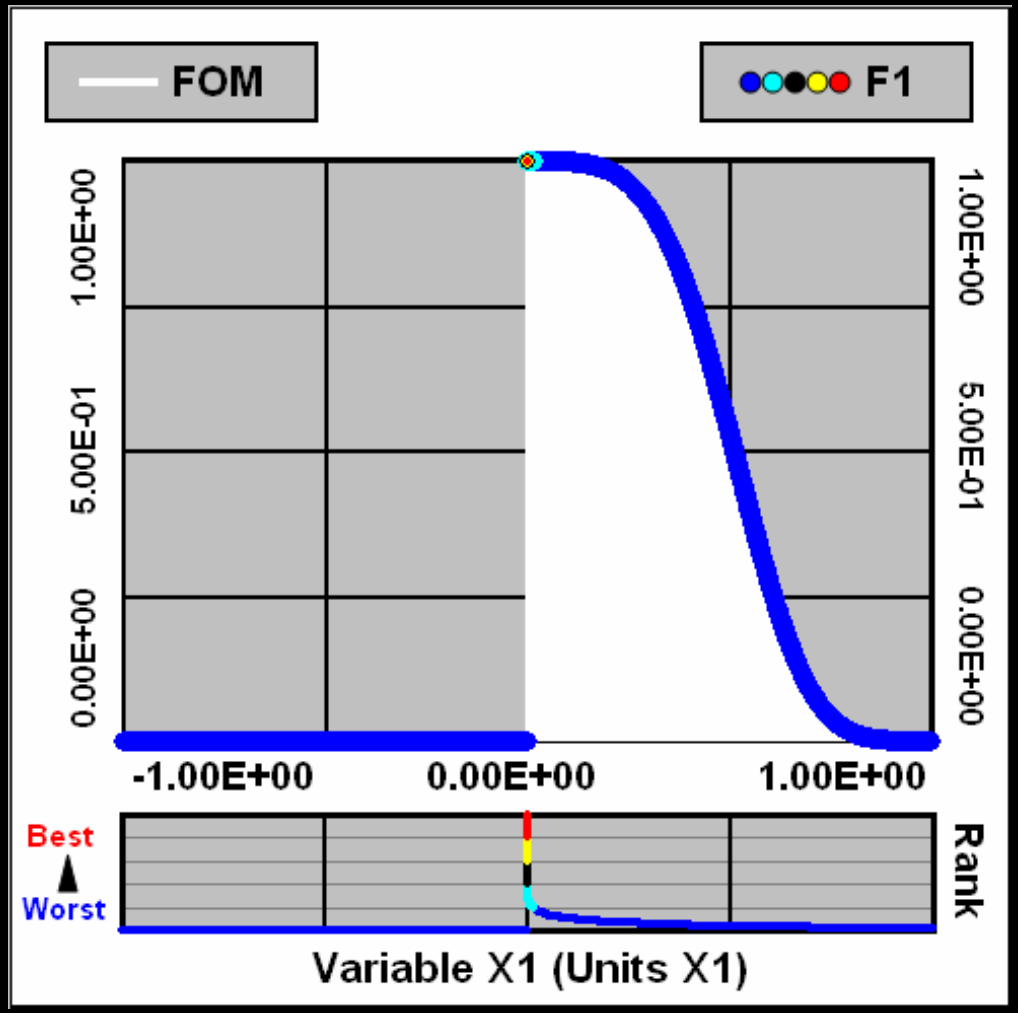
The PSE algorithm does not require functions to be smooth and differentiable, or even continuous.

Example: Maximize the function

F1(X1) = IF[X1<0, 0, Exp(-10*(X1^4))]

In a sense, this is one of the most difficult problems for an optimization routine to handle because the best solution lies infinitesimally close to the worst solution.

We'll solve it using a particle population size Np = 2 (the Pod size for 1-D problems)



Optimizing Discontinuous Functions

Andromeda

F1=IF(X1<0,0,EXP(-10*(X1^4)))

Functions		Goal	Target	Wt	Best Value	Error
Function F1 (Units F1)	F1	F1 =	1.00E+00	1	1.00000000000000E+00	0.E+00

Input Variables		Min	Max	B	Best Value	± Range
Variable X1 (Units X1)	X1	-1.00E+00	1.00E+00	Y	3.25646406961869E-05	3.E-05

Termination: Specified Function Error Tolerance (Zero Error) Achieved

# Runs (Iterations) =	39
Particle Population (Np) =	2
Function Evals =	78
Hard Constraint Evals =	0

Execution Time Components (sec)	
Function Evals =	0.02
PSE Algorithm =	0.02
Total RunTime =	0.33

Search Mode
Rapid Convergence
Memory Used
38.00 MB

The solution is found in 39 iterations (78 function evaluations)

"Hard" vs "Soft" Constraints

A constraint is any additional condition that the solution must satisfy.

"Hard" constraint
One that must be exactly satisfied (within the precision limits of the machine).

"Soft" constraint
One that is desired to be satisfied, and pushes the solution in the direction of satisfying the condition expressed by the constraint, but is not guaranteed to be exactly satisfied. The degree to which the constraint is desired to be satisfied (the amount of "pushing") is given by its Weight (Wt) , which is assigned by the user.

Soft constraints are more efficient to work with than Hard constraints, but there are times when Hard constraints must be used (such as when the expression for an input function is not valid unless the constraint is satisfied).

As we will demonstrate on the next few slides, soft constraints work by modifying the FOM (Figure Of Merit) which rank orders the solutions to the optimization problem. The mathematical definition of the FOM is presented in the Appendix of this user guide.

"Hard" vs "Soft" Constraints

When constraints are activated on the "Run" worksheet, they are color coded by Type:

Hard = Cyan
Soft = Pink

Constraints

C's	Enter Equation	Select Constraint GOAL	Select Constraint Type	Wt.
C1	-4.44E-16	= 0	Soft	1
C2	-5.00E-01	< 0	Hard	1
C3	0.00E+00	> 0		1
C4				1
C5				1
C6				1
C7				1

Andromeda

F1=SIN(PI()*SQRT(X1^2+X2^2))

Functions		Goal	Target	Wt	Best Value	Error
Function1 (Units F1)	F1	Max	1.00E+00	1	1.00000000000000E+00	0.E+00

Constraints		Goal	Type	Wt	Equation
	C1	C1 = 0	Soft	1	C1=X1-X2
	C2	C2 < 0	Hard	INF	C2=SQRT(X1^2+X2^2)-1

Input Variables		Min	Max	B	Best Value	± Range
Variable X1 (Units X1)	X1	-1.00E+00	1.00E+00	Y	3.53553391996777E-01	1.E-08
Variable X2 (Units X2)	X2	-1.00E+00	1.00E+00	Y	3.53553391996777E-01	1.E-08

Termination: Specified Function Error Tolerance (Zero Error) Achieved

All active constraints are reported on the Results Summary

Example: Soft Constraint "X1 > 0"

Example: Maximize
F1(X1) = Sin[pi()*X1^2]^2
over the interval |X1| ≤ 2
with Soft constraint X1>0

Enter the constraint
- Equation
- Goal
- Type

as shown here.

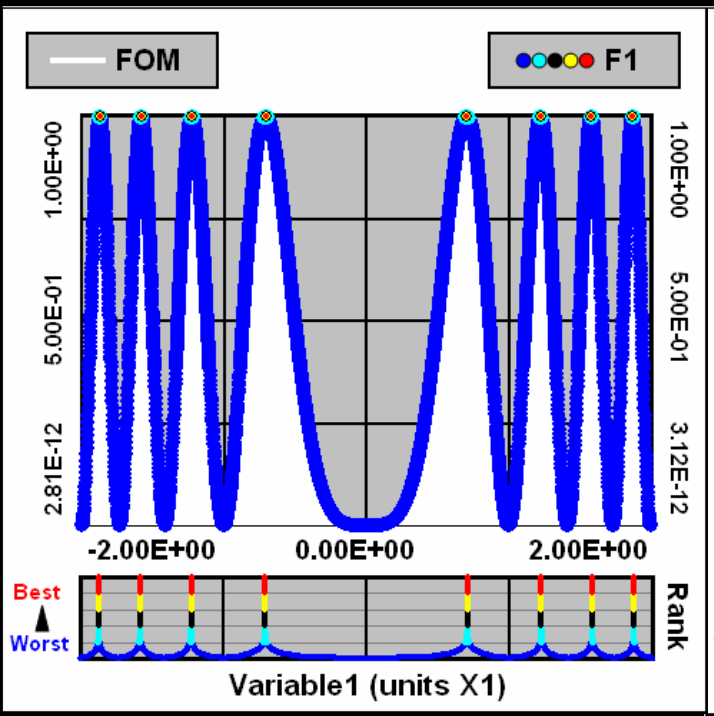
Constraints

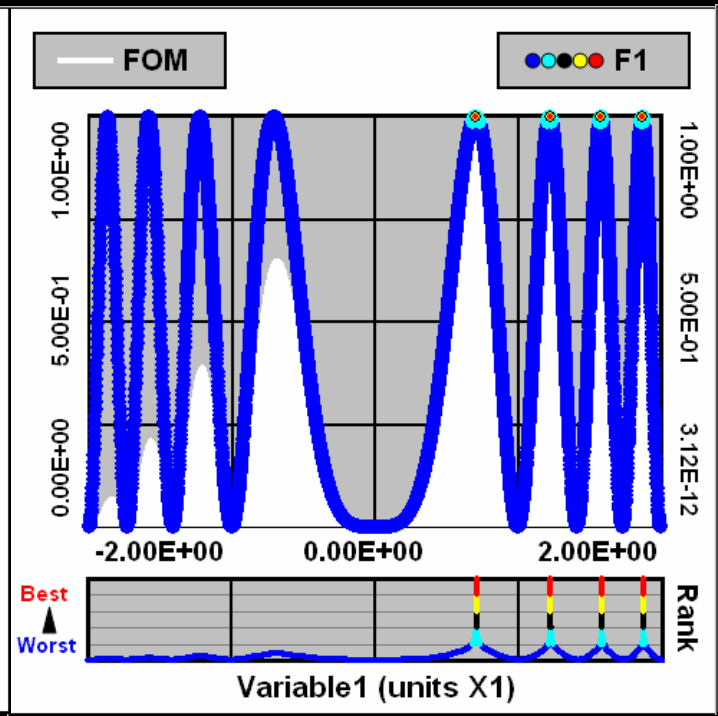
C's	Enter Equation	Select Constraint GOAL	Select Constraint Type	Wt.
C1	=X1	> 0	Soft	1
C2				1
C3				1
C4				1
C5				1
C6				1
C7				1

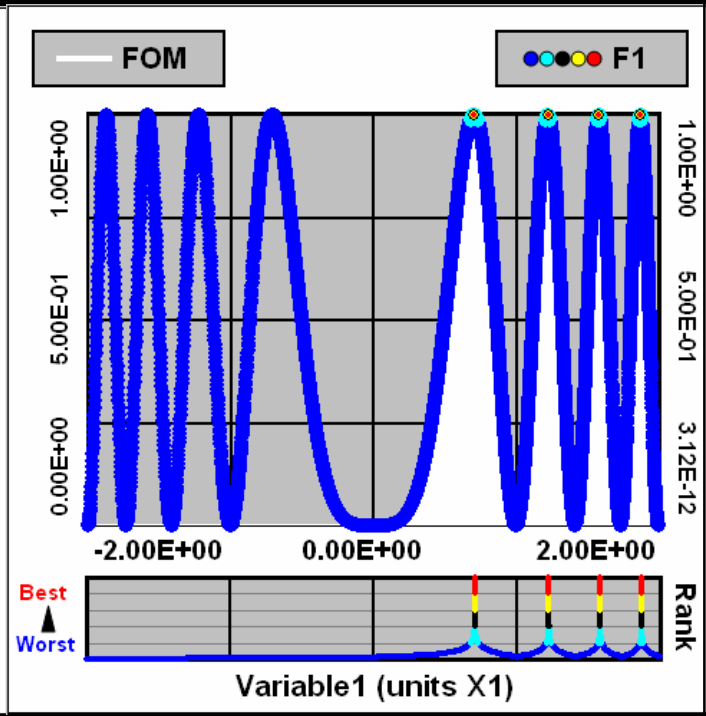
The results obtained for different Weight values are shown on the next slide.

Example: Soft Constraint "X1 > 0"

This Soft constraint penalizes the FOM (Figure Of Merit, plotted in white) for all negative values of X1, and the penalty increases as X1 becomes more negative.







Wt = 0

Equivalent to no constraint. The solutions for X1 can be positive or negative.

Wt = 1

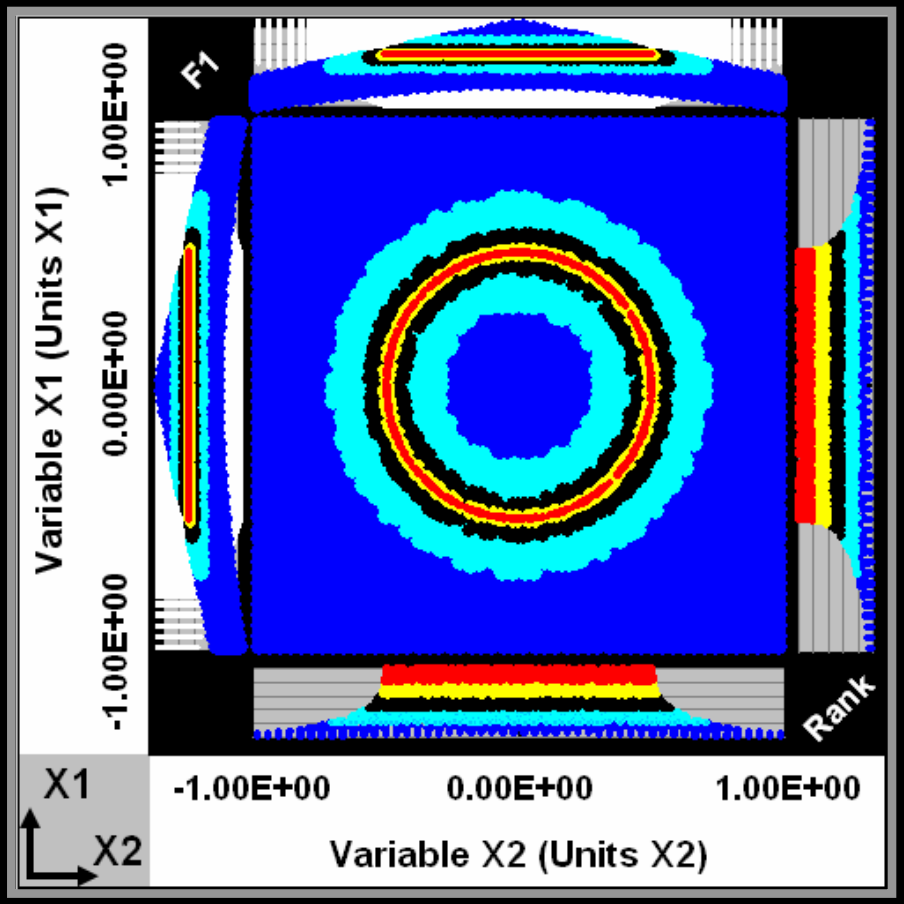
The FOM is penalized (reduced) for negative values of X1, and the particles converge on positive solutions for X1.

Wt = 100

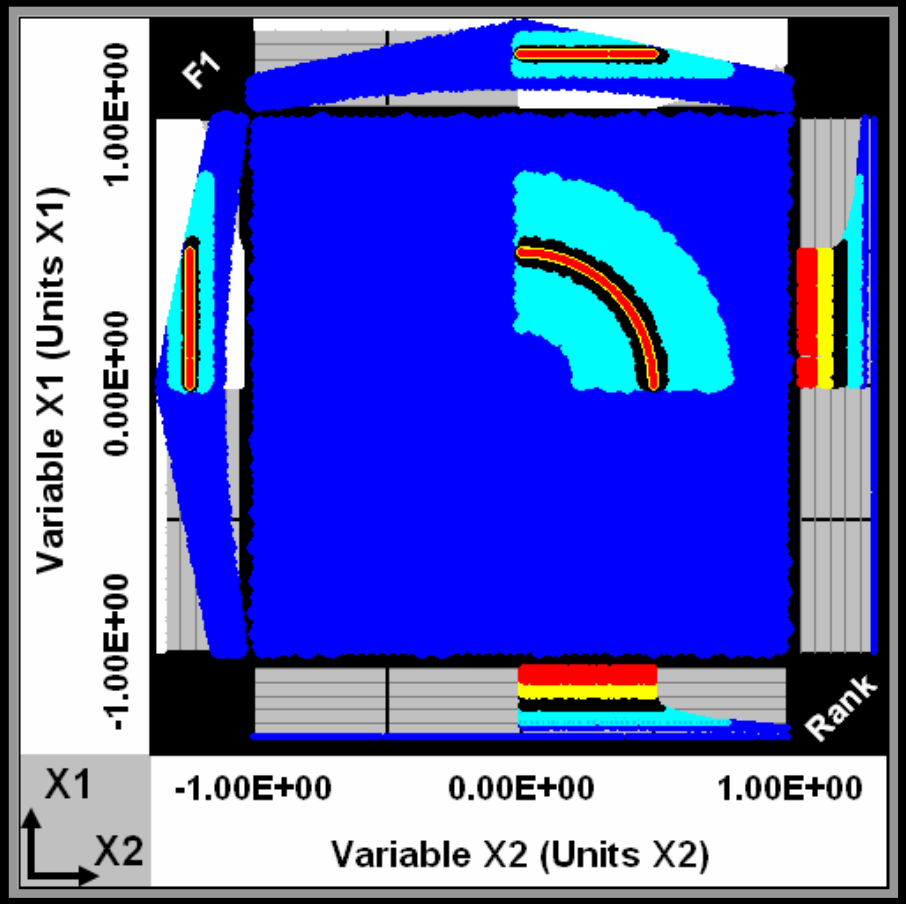
The penalty is now so large that the FOM is essentially zero for all negative values of X1.

Example: A Soft Constraint in 2-D

Solve: F1(X1,X2) = SQRT(X1^2+X2^2) = 0.5
with Soft Constraint "AND(X1>0, X2>0)"



Wt = 0
(no constraint)



Wt = 1

"Rapid" vs "Thorough" Search Mode

The two different Search Modes of the PSE algorithm were previously described in Chapter II, and now we'll take a look at some examples to explore the relative merits of each.

Search Mode 1: "Rapid"

This mode offers the most rapid convergence to solutions because only the best particles are propagated in each iteration of the PSE algorithm.

Search Mode 2: "Thorough"

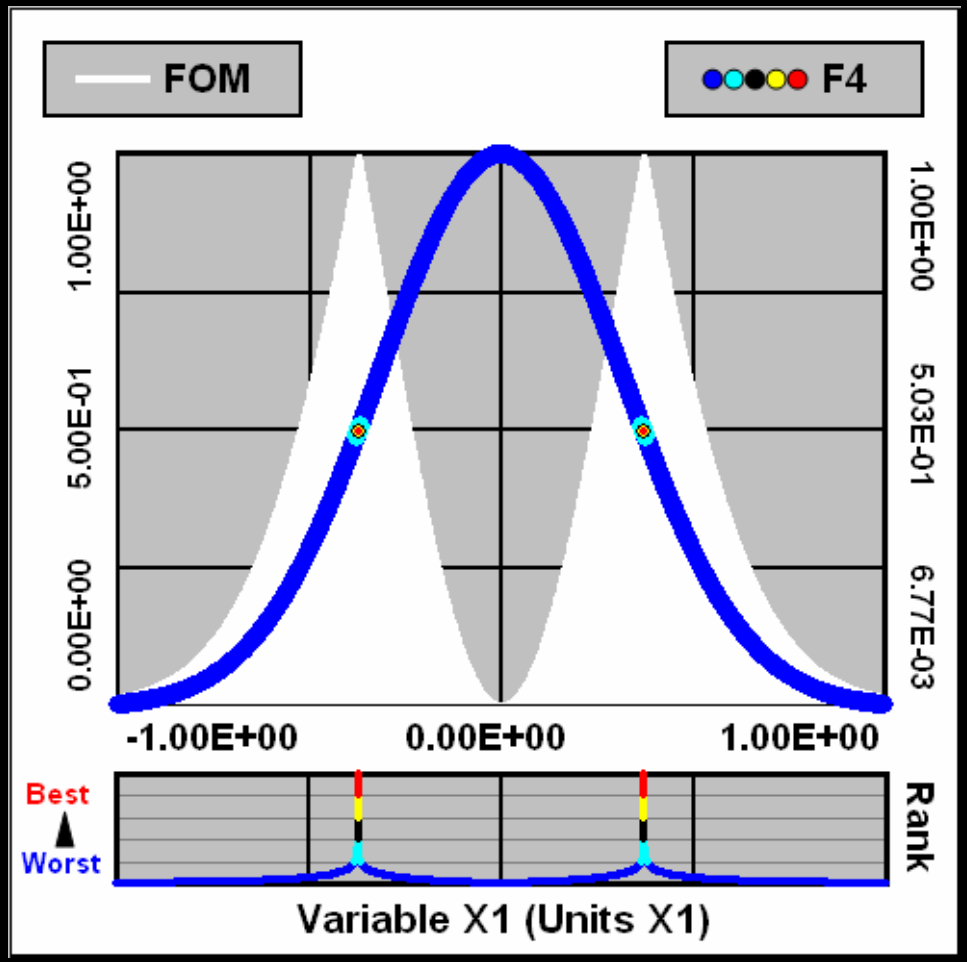
In this mode, the particles will never cease exploring any local optimum once convergence has begun, and therefore this mode provides the most thorough search of local optima that is possible.

"Rapid" vs "Thorough" Search Mode

Example: We will solve

$$F(X1) = Exp[\ -5*(X1^2)\ ] = 0.5$$

over the interval $|X1| \leq 1$.

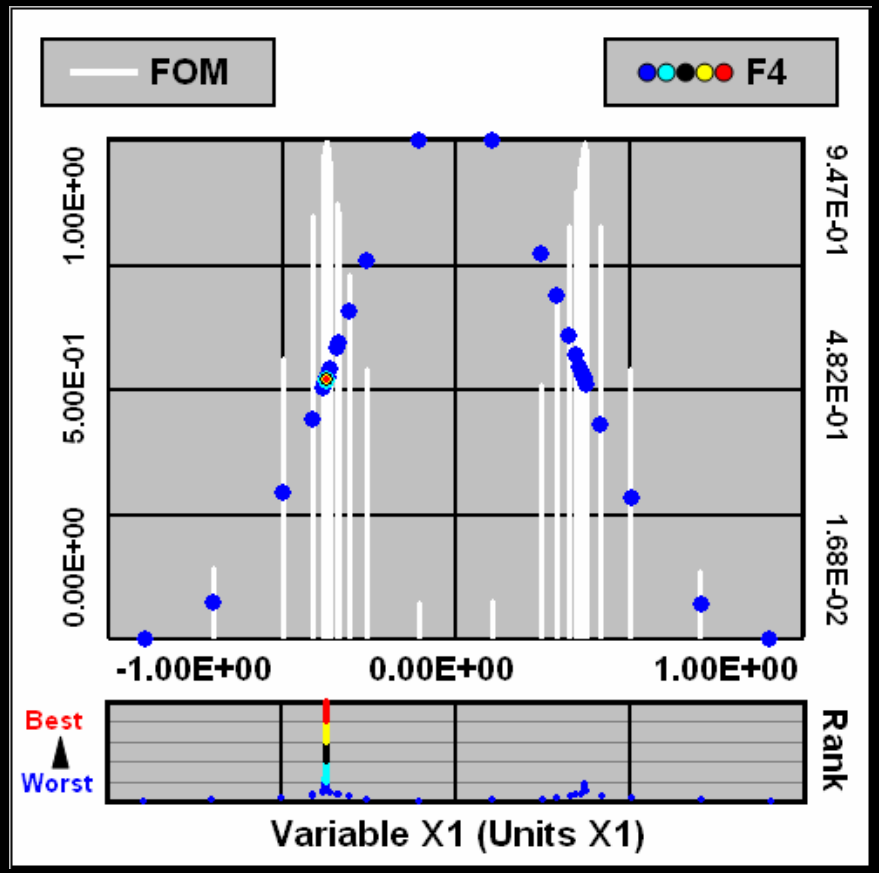


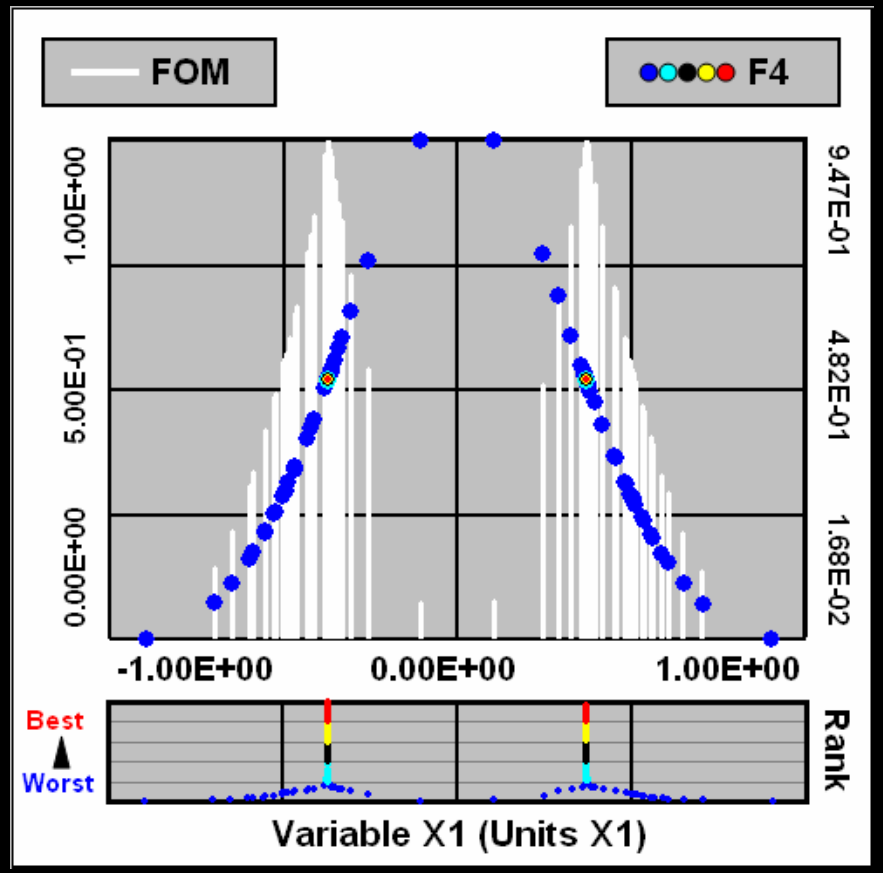
There are two equally good solutions to this problem.

"Rapid" vs "Thorough" Search Mode

$$F(X1) = Exp[\ -5*(X1^2)\ ] = 0.5$$

Here are the results we obtain using Np = 10 particles and the two different Search Modes.





Search Mode = "Rapid"

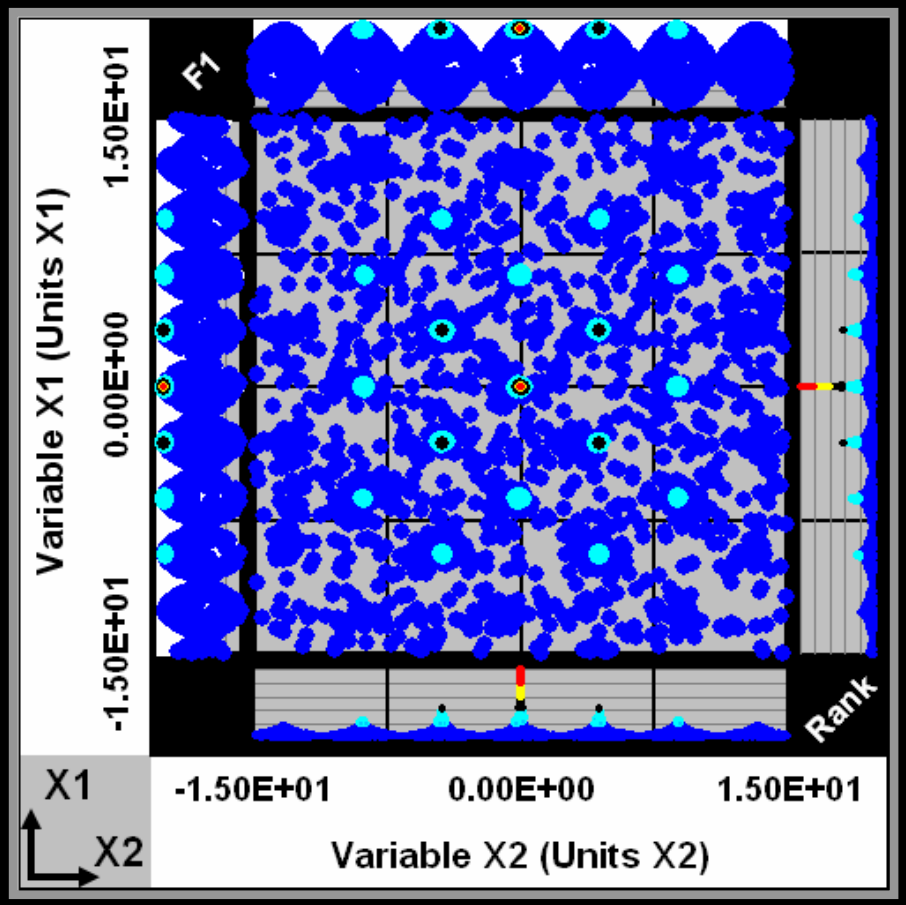
Convergence eventually shifts to just one of the two equivalent solutions.

Search Mode = "Thorough"

Once particles begin converging on either solution, they continue exploring that solution in all subsequent iterations.

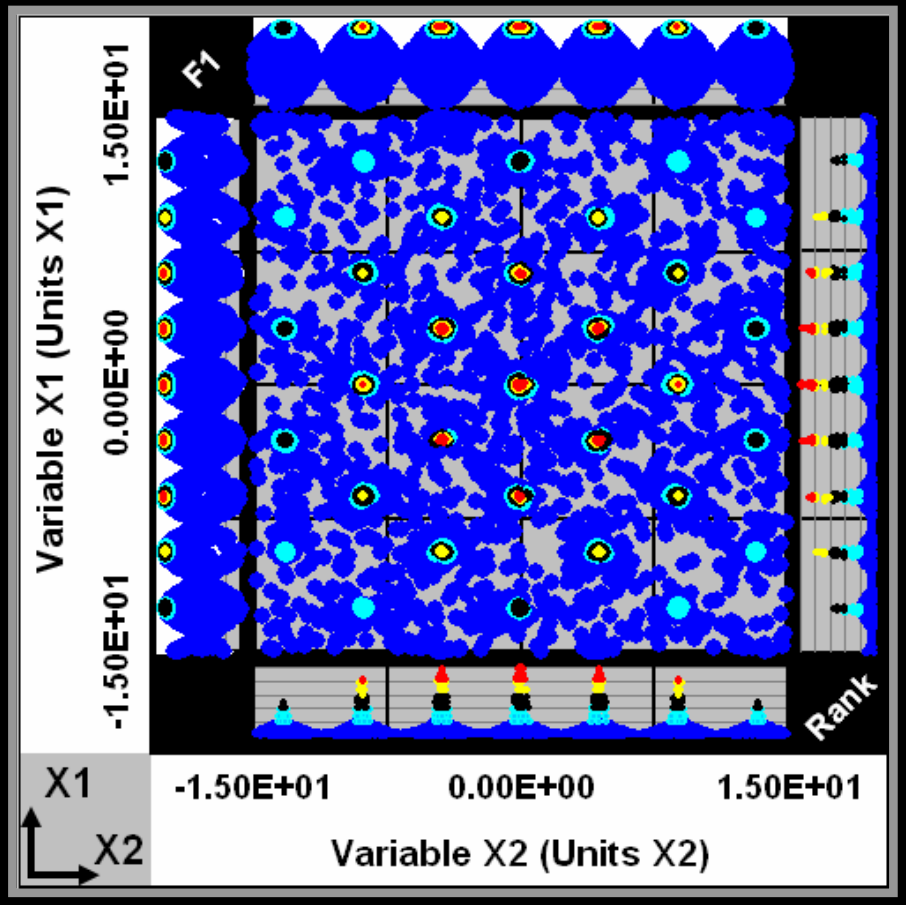
"Rapid" vs "Thorough" Search Mode

Example: Minimizing Griewangk's Function (2-D)



Search Mode = "Rapid"

Particle convergence focuses on the best local optima.



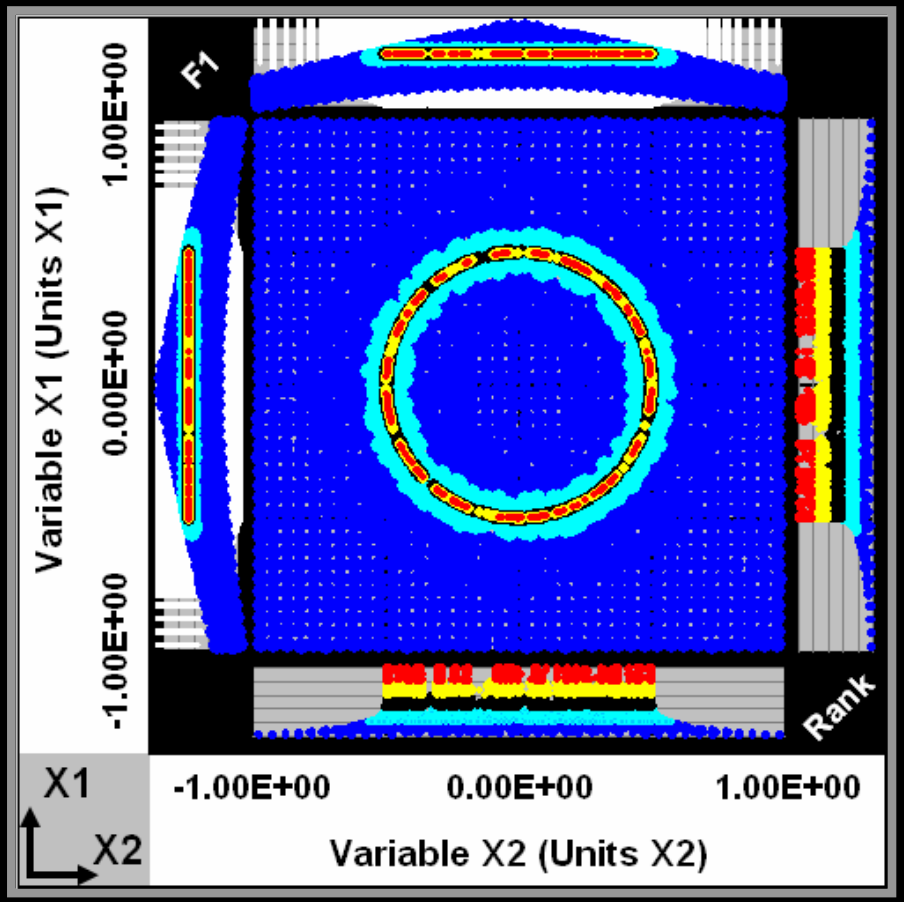
Search Mode = "Thorough"

The particles continue exploring all of the local optima, regardless of the number of iterations.

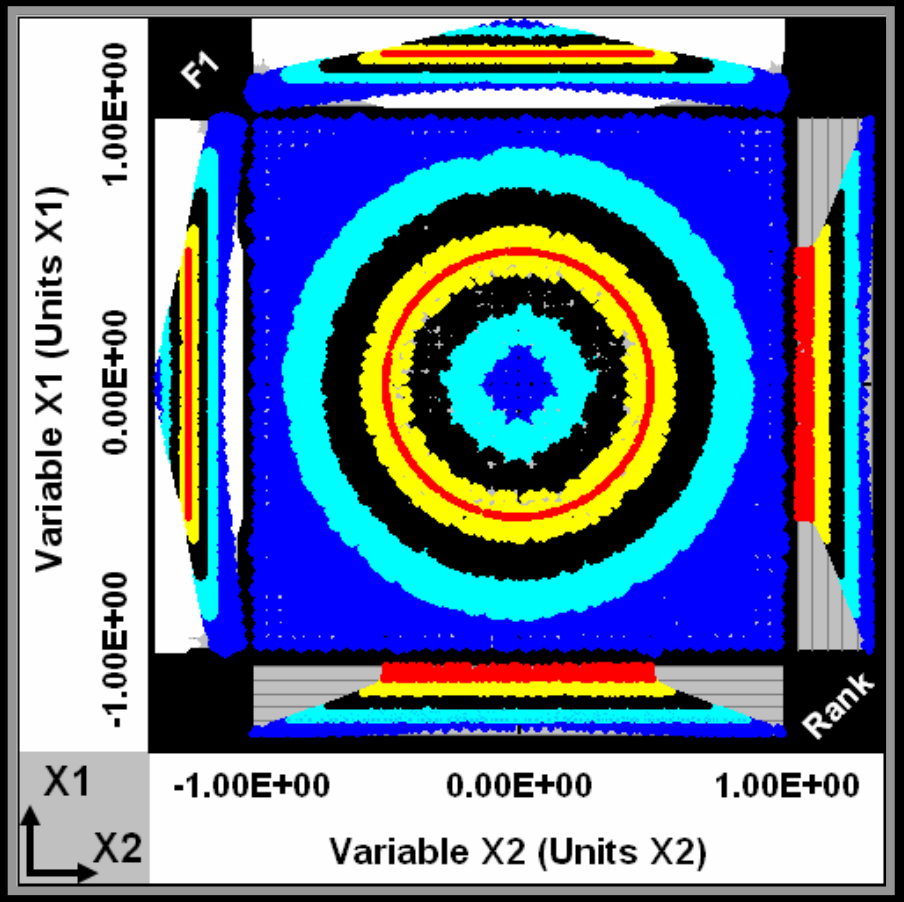
"Rapid" vs "Thorough" Search Mode

Example: F1(X1,X2) = SQRT(X1^2+X2^2) = 0.5

Np = 1600 Particles, # Iterations = 10



Search Mode = "Rapid"



Search Mode = "Thorough"

We see that the particles have converged much more rapidly to individual solutions in the "Rapid" search mode, whereas the particles are doing a more thorough job of exploring all the equivalent solutions in the "Thorough" search mode.

"Rapid" vs "Thorough" Search Mode

Suggested Uses of each Search Mode:

The "Rapid" search mode is the normal mode of operation, and is sufficiently thorough in exploring all local optima whenever the particle population size is sufficiently large, that is:

$$Np >> (\text{Pod Size}) \times (\#\ \text{Local Optima})$$

If that condition is not met, or if you wish to remain aware of all the local optima positions within the search space, then one should use the "Thorough" search mode. The "Thorough" search mode is the preferred mode of operation when first solving a problem for which the number of local optima is unknown.

In either case, using a large particle population Np with a small number of iterations (5-10) and then examining the Contour Matrix Plot is always a good first step in understanding the optimization problem that you're trying to solve.

Multiple Function Optimization

All problems in the optimization of multiple objectives (e.g., F1= Max AND F2= Max) will fall into one of the following two categories:

1) Trade-Off Optimization

We define this as the case where two or more of the objectives can not simultaneously be achieved, and therefore a trade-off among the objectives must be made. The global solution(s) in this case will be a function of the weight values (Wt) that the user specifies for each function objective.

2) Simultaneous Optimization

We define this as the case where all objectives can simultaneously be achieved. The global solution(s) in this case will NOT depend upon the weight values (Wt) that the user specifies for each function objective.

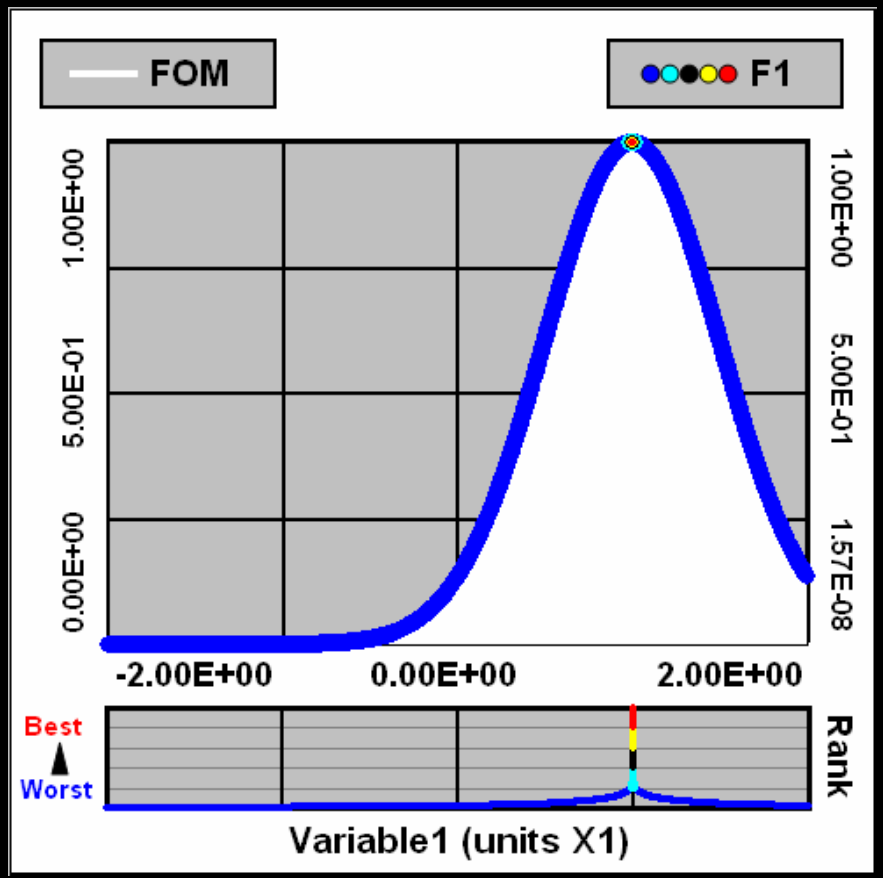
We'll present the following examples in this section:

1-D Trade-Off

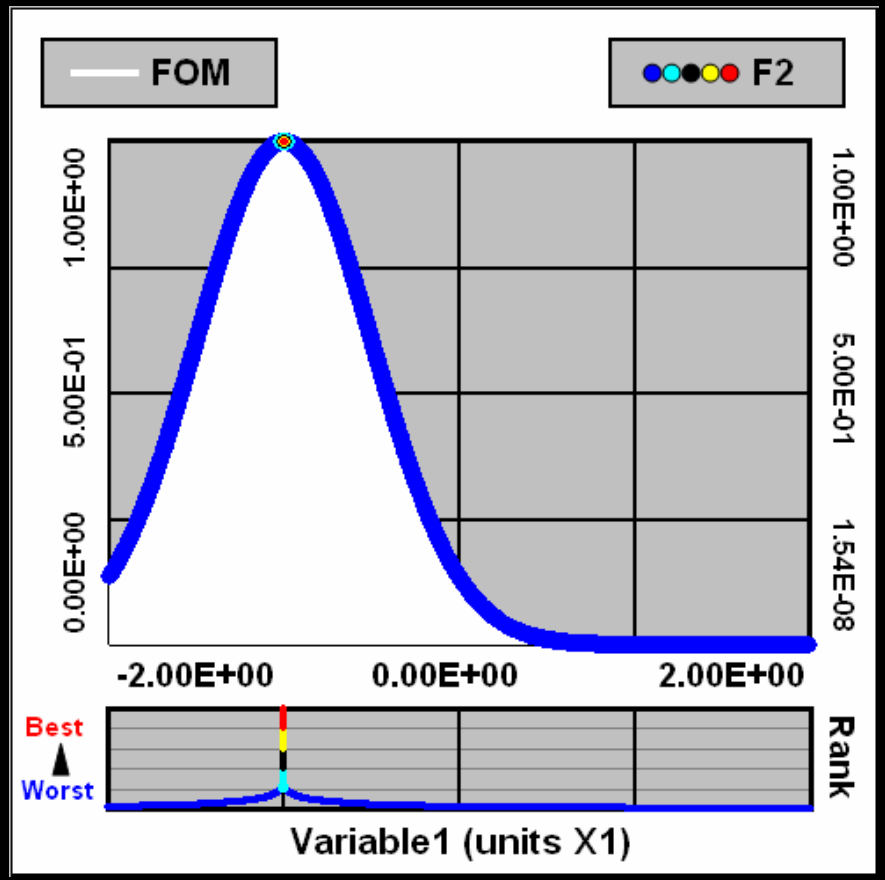
Solve: F1(X1) = Exp(-2*(X1-1)^2) = Max
AND
F2(X1) = Exp(-2*(X1+1)^2) = Max

All problems in the optimization of multiple objectives begin with the following first step:

Step #1: Optimize each Function individually to find the best results possible.



Max F1(X1) = 1



Max F2(X1) = 1

1-D Trade-Off

Solve: F1(X1) = Exp(-2*(X1-1)^2) = Max
AND
F2(X1) = Exp(-2*(X1+1)^2) = Max

Step #2: Regardless of whether the Goal is to Maximize or Minimize a function, or to set it equal to some particular value, enter the best result you obtained for each function as the "Target" value:

Functions

F's	Enter Equation	Select Function GOAL	Enter Target Value	Wt.	Name (Units)
F1	1.35E-01	Max	1.00E+00	1	Function1 (Units F1)
F2	1.35E-01	Max	1.00E+00	1	Function2 (Units F2)
F3				1	
F4				1	
F5				1	

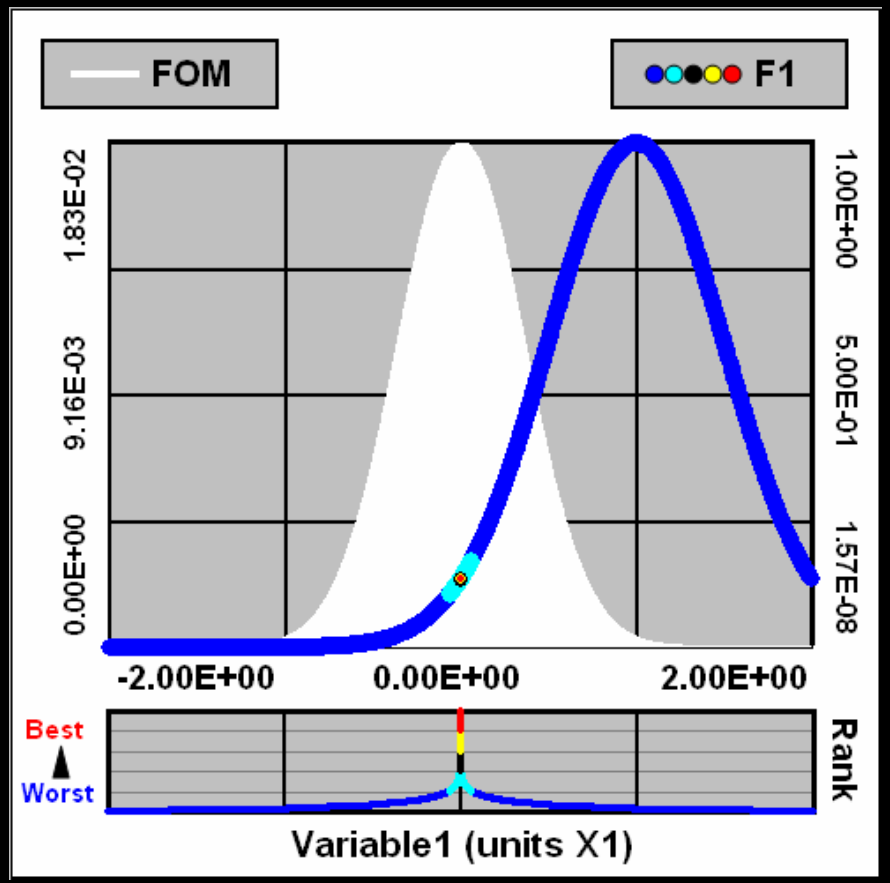
A weight value $Wt \geq 0$ representing the relative importance of each objective must be entered.

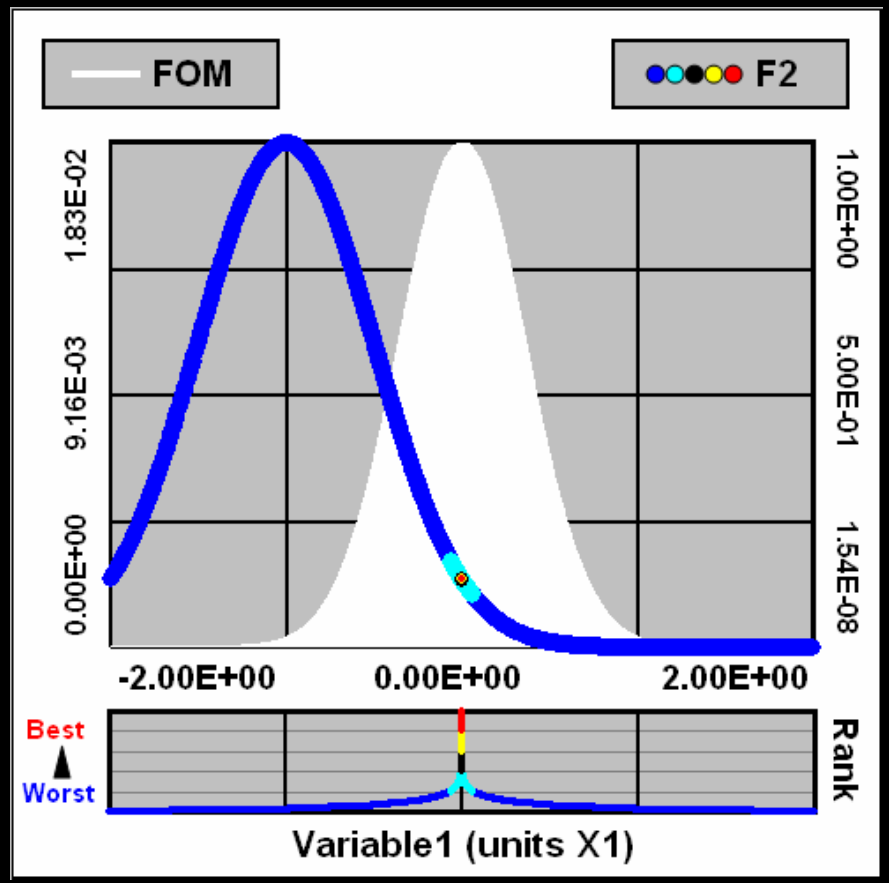
When optimizing multiple functions, you must enter a Target value for each function representing the best result that can be achieved when each function is optimized separately. (High precision is not required)
Those values are used to construct the FOM (Figure Of Merit) .

1-D Trade-Off

Solve: F1(X1) = Exp(-2*(X1-1)^2) = Max
AND
F2(X1) = Exp(-2*(X1+1)^2) = Max

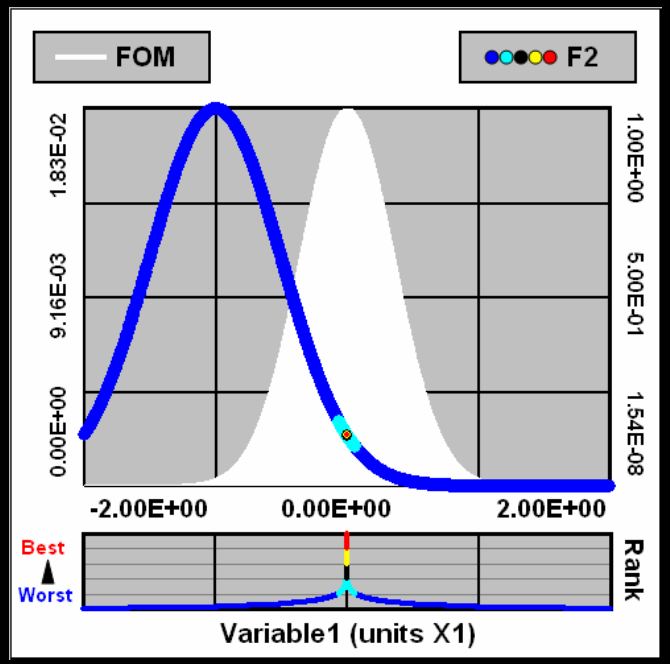
We then obtain the following solution when the weight for each objective is Wt=1. Notice that the FOM (plotted in white) is maximized midway between the maxima of F1 and F2.



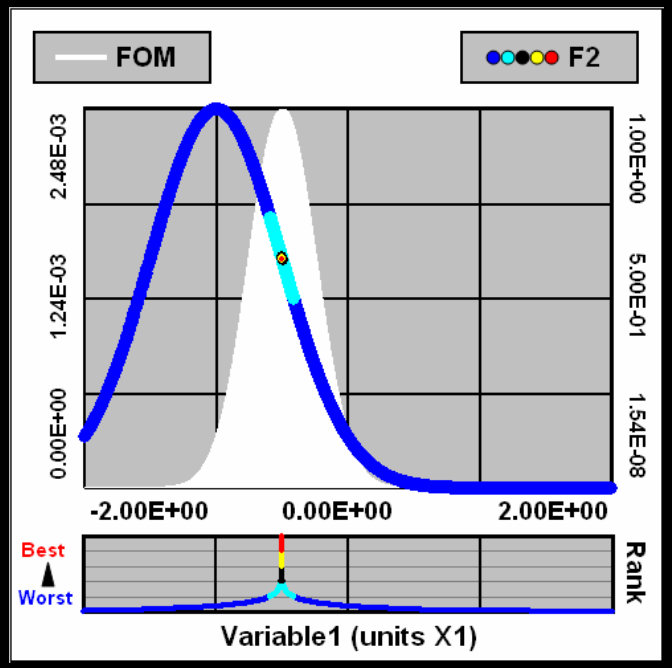


1-D Trade-Off

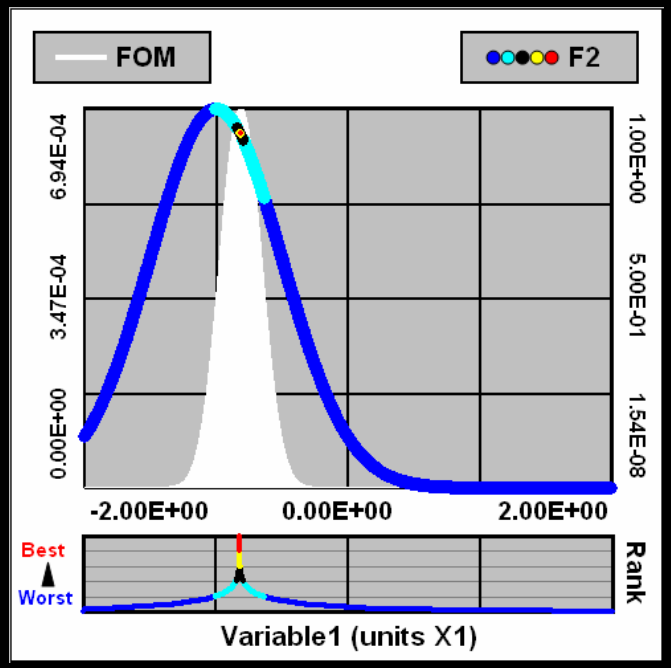
Increasing the Weight (Wt) for F2 relative to F1 will shift the FOM to solutions that increasingly favor F2.



F1 Wt = 1
F2 Wt = 1



F1 Wt = 1
F2 Wt = 3



F1 Wt = 1
F2 Wt = 10

1-D Trade-Off

An Alternative Approach: A "Cost" Function for Multiple Objectives

A Trade-Off analysis such as our 1-D example in which the global solution is determined by the weight factors representing the relative importance of each objective is obviously very qualitative.

In order to be able to rigorously trade-off different objectives in multi-objective optimization, one requires an accurate "Cost" function that quantitatively expresses the impact (in whatever way one chooses to define it) of not meeting each objective. For example, if we seek to Maximize F1 and F2, a linear Cost model might be something of the form:

$$Cost(F1,F2) = CF_1(1 - F1/F1max) + CF_2(1 - F2/F2max)$$

where the CF_i are fixed parameters which represent the cost of each function Fi being less than its maximum value.

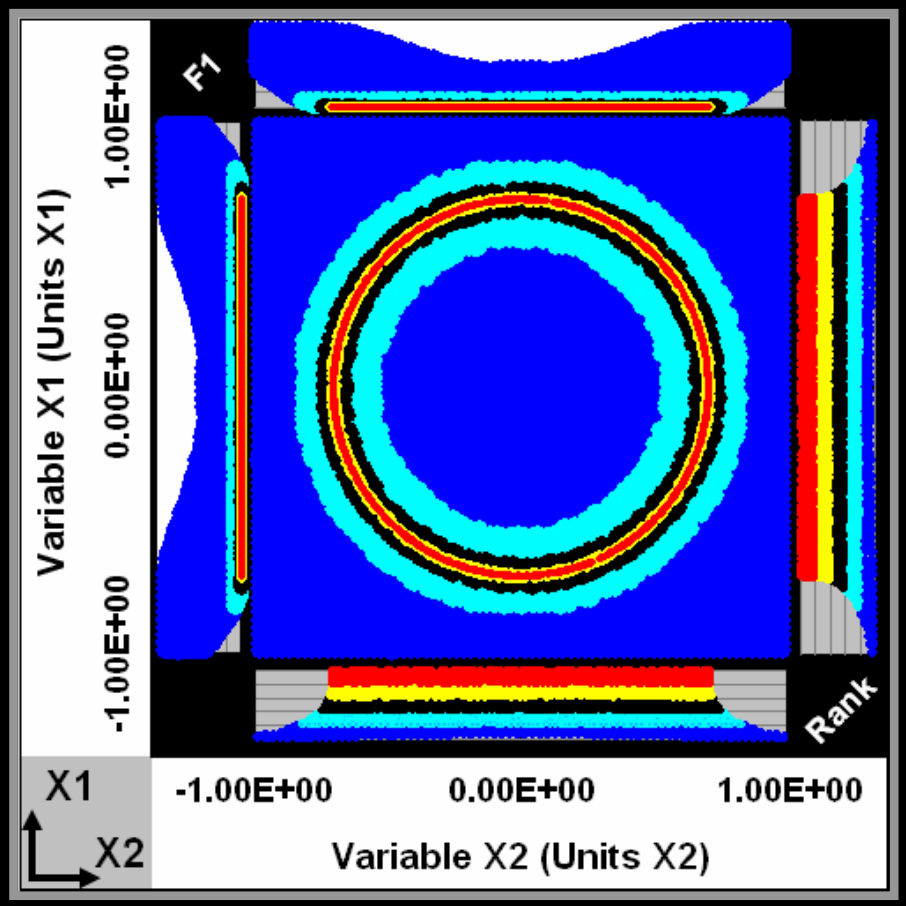
The multi-objective problem then reduces to a single objective we need to solve for:

$$Cost(F1,F2) = Min$$

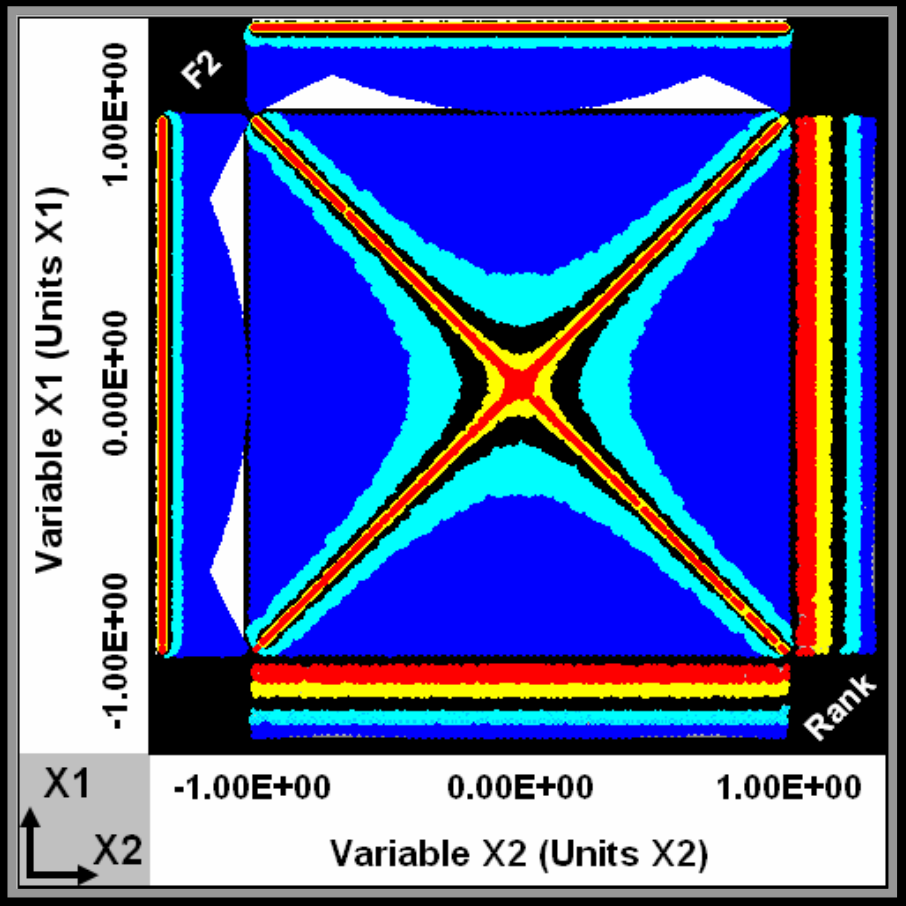
2-D Simultaneous

Solve: F1(X1,X2) = Sin[π(X1^2+X2^2)] = Max
AND
F2(X1,X2) = Abs(X2^2-X1^2) = Min
for $|X_i| \leq 1$

All problems in the optimization of multiple objectives begin with the following first step:
Step #1: Optimize each Function individually to find the best results possible.



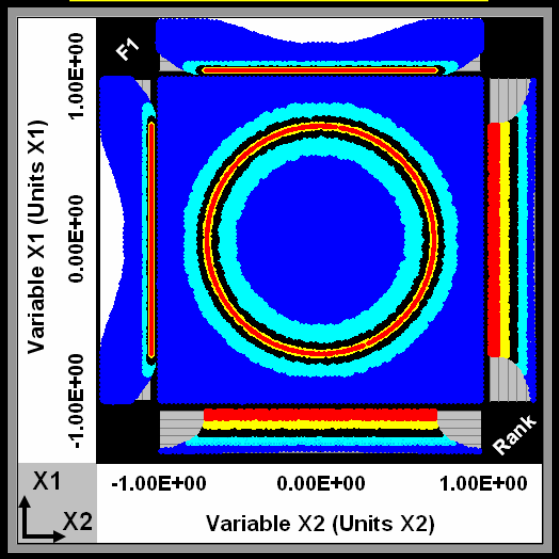
Max F1(X1,X2) = 1



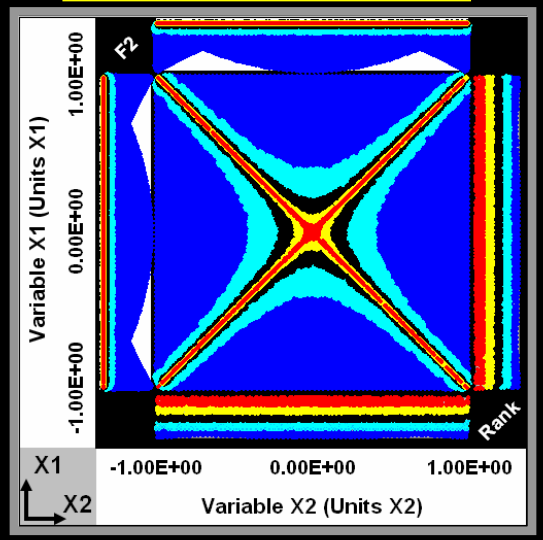
Min F2(X1,X2) = 0

2-D Simultaneous

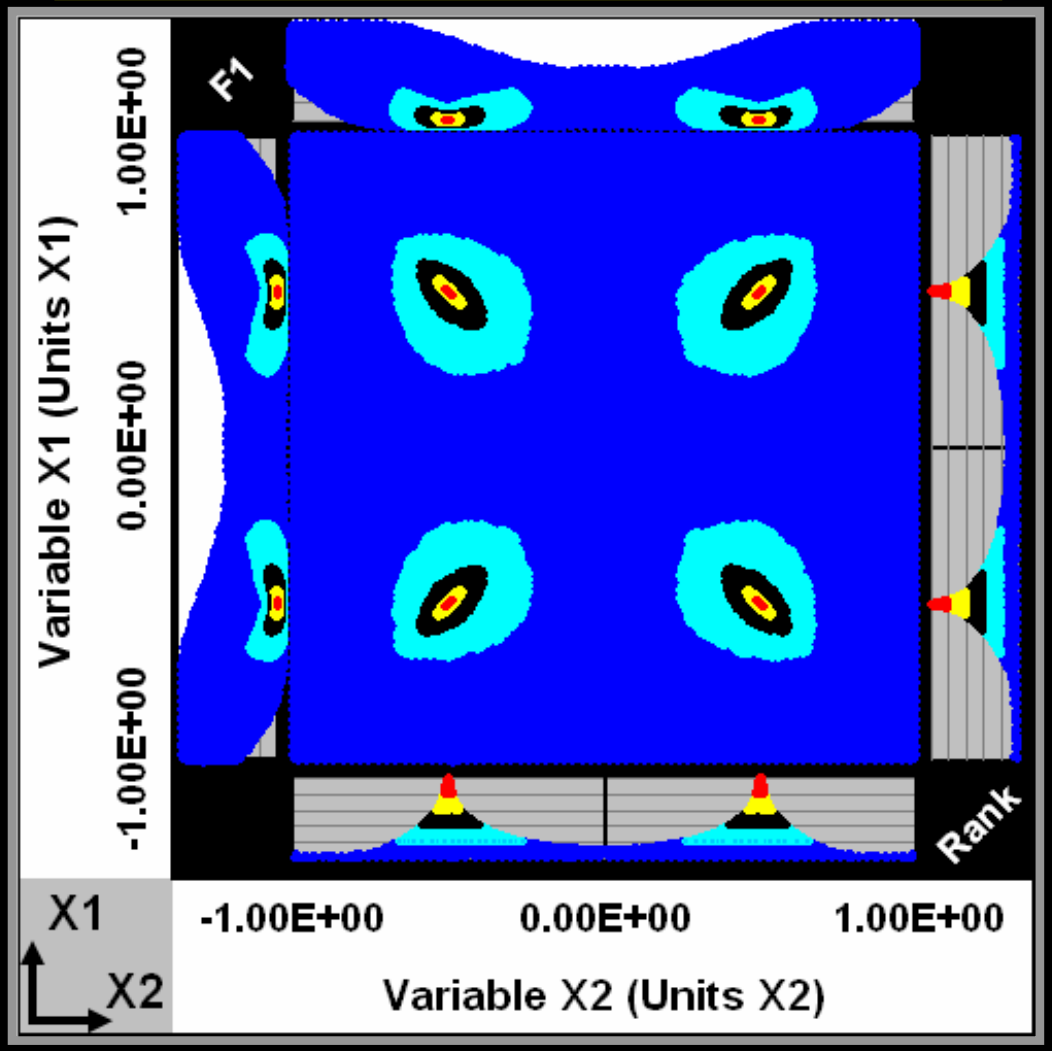
F1(X1,X2) = Max



F2(X1,X2) = Min



F1(X1,X2) = Max AND F2(X1,X2) = Min

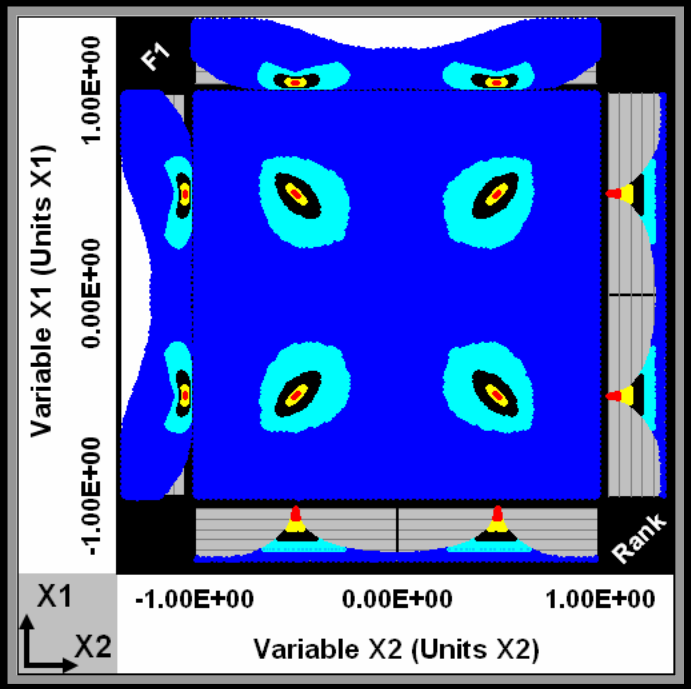


We find the following global solutions:

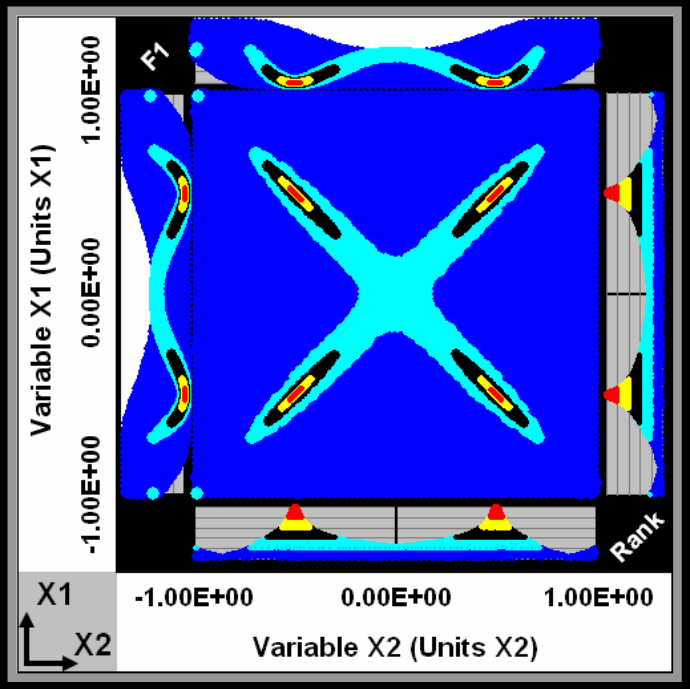
$X_i = \pm 0.5$

2-D Simultaneous

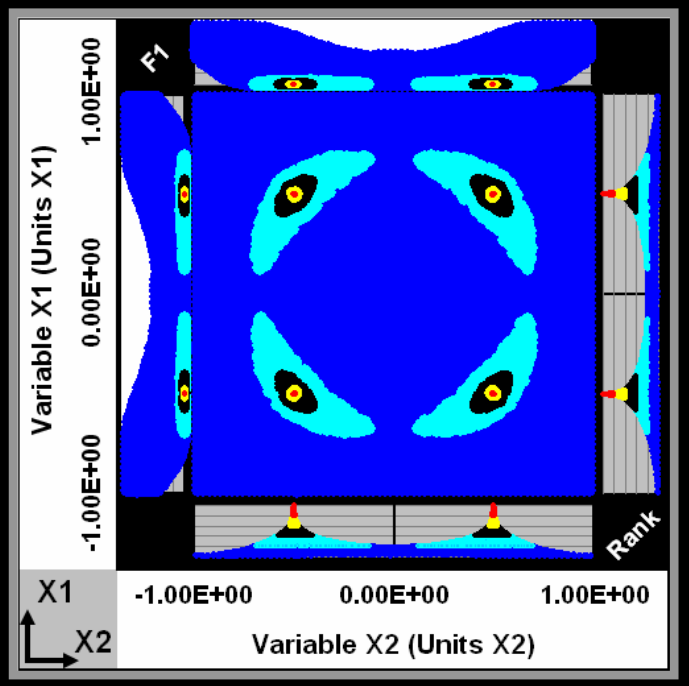
**Whenever simultaneous solution is possible,
changing the Weight values for F1 and F2 can impact the shape of the particle distributions,
but will NOT change the global solution values
(which in this case are: $X_i = \pm 0.5$)**



F1 Wt = 1
F2 Wt = 1



F1 Wt = 1
F2 Wt = 10

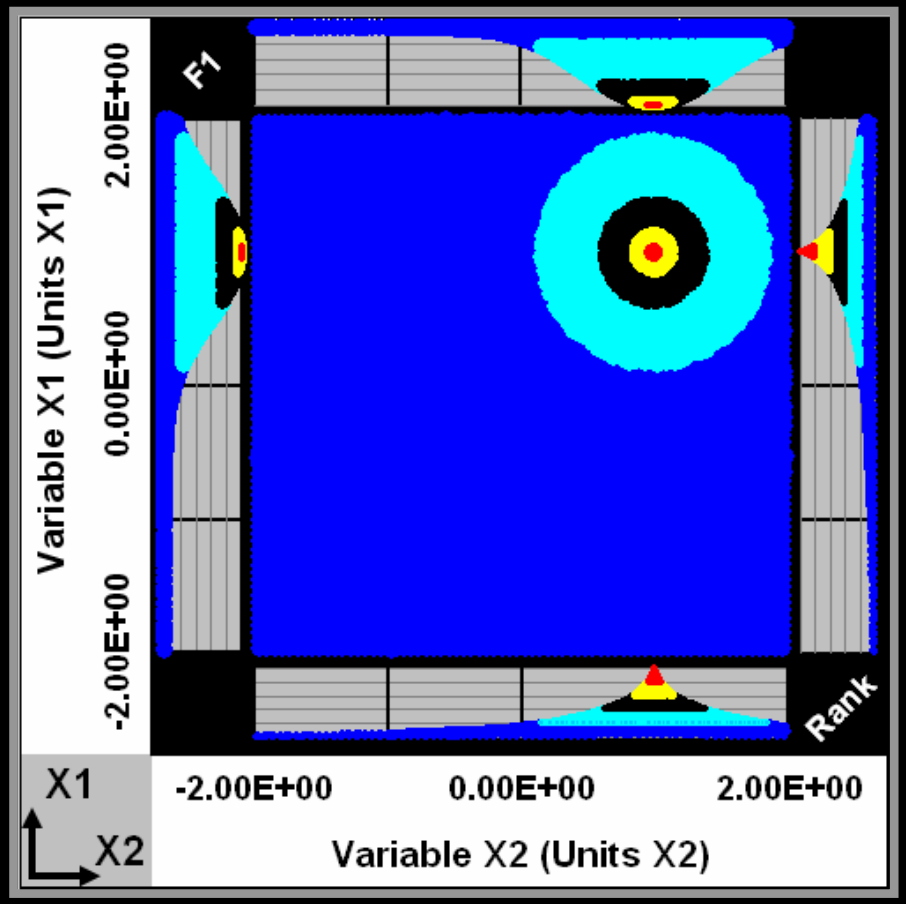


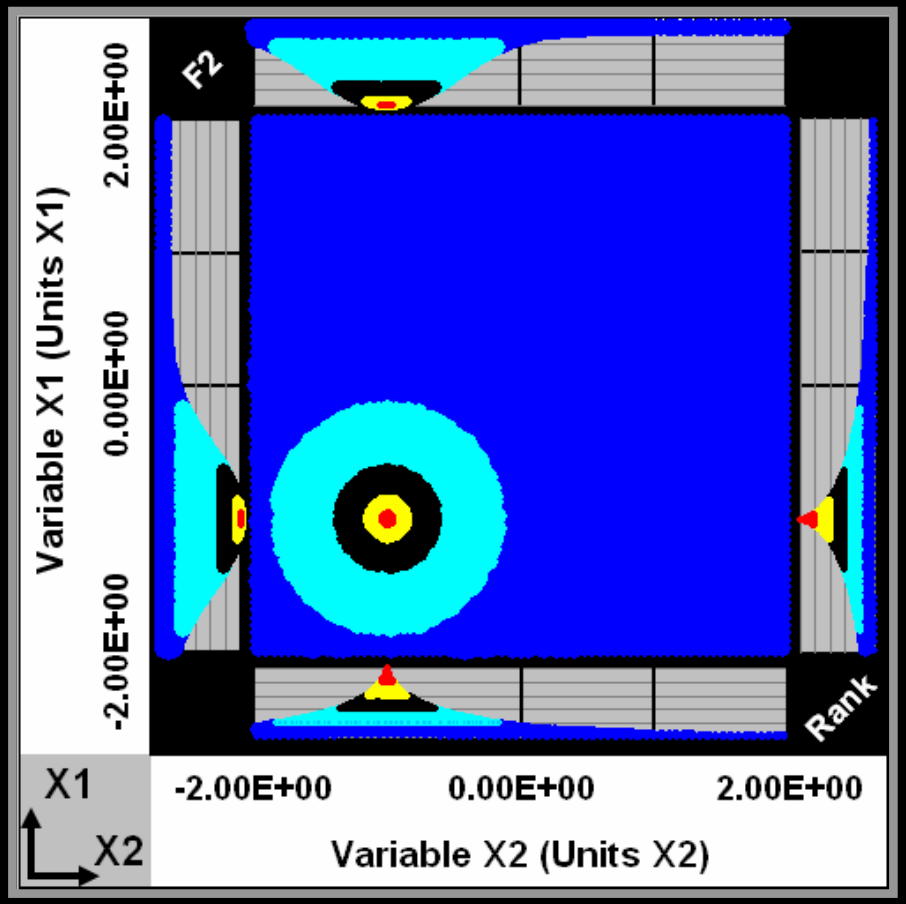
F1 Wt = 10
F2 Wt = 1

2-D Trade-Off

Solve: F1(X1,X2) = Exp(-2*[(X1-1)^2+(X2-1)^2]) = Max
AND
F2(X1,X2) = Exp(-2*[(X1+1)^2+(X2+1)^2]) = Max

All problems in the optimization of multiple objectives begin with the following first step:
Step #1: Optimize each Function individually to find the best results possible.



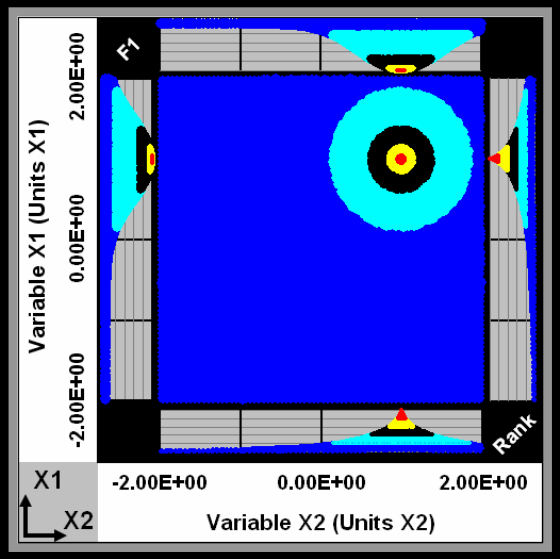


Max F1(X1,X2) = 1

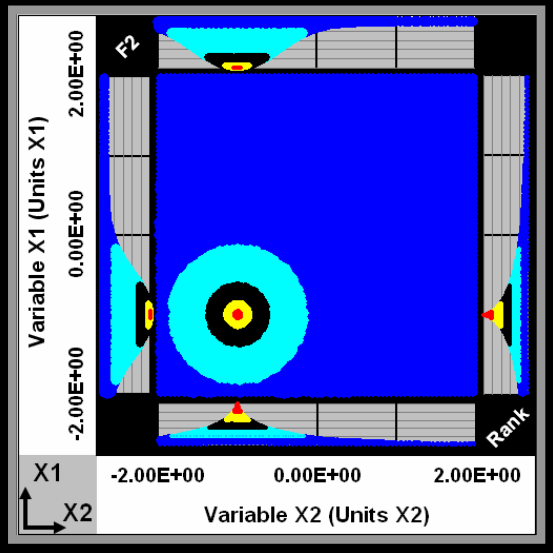
Max F2(X1,X2) = 1

2-D Trade-Off

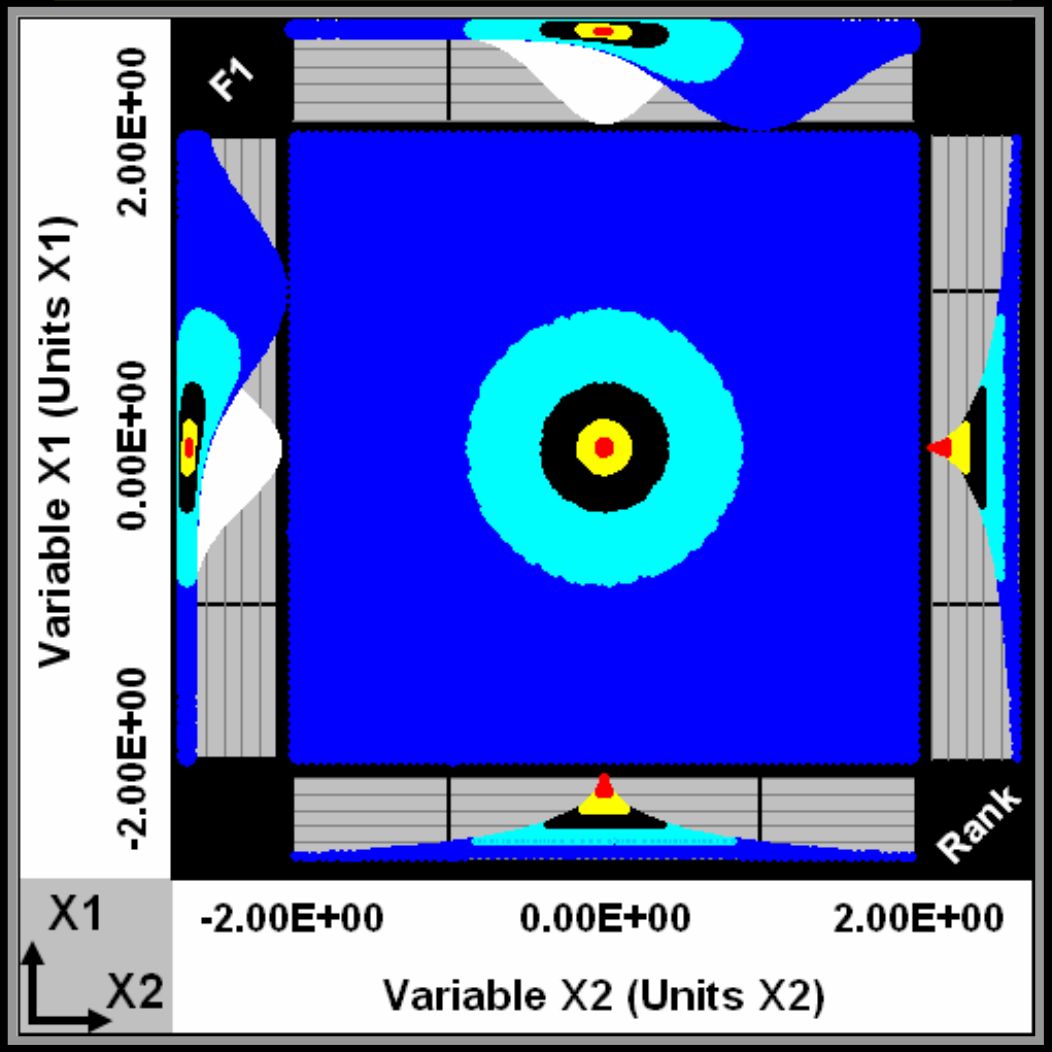
F1(X1,X2) = Max



F2(X1,X2) = Max



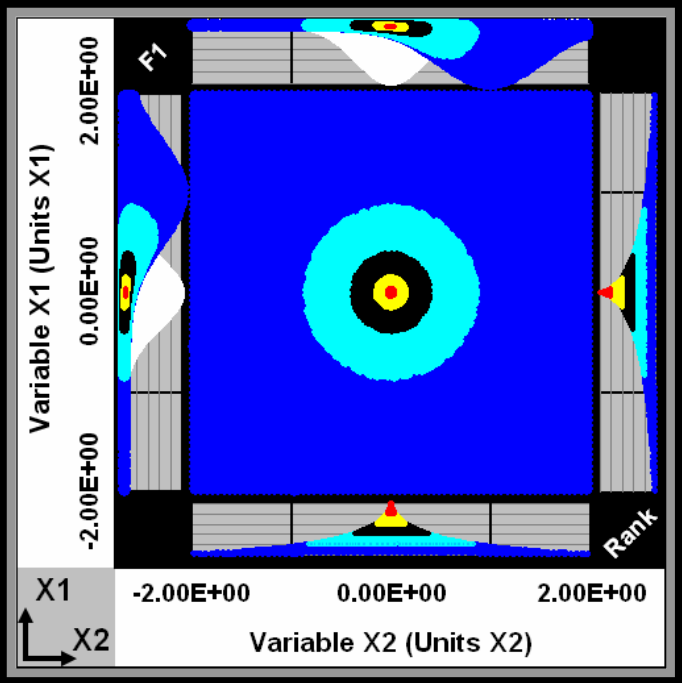
F1(X1,X2) = Max AND F2(X1,X2) = Max



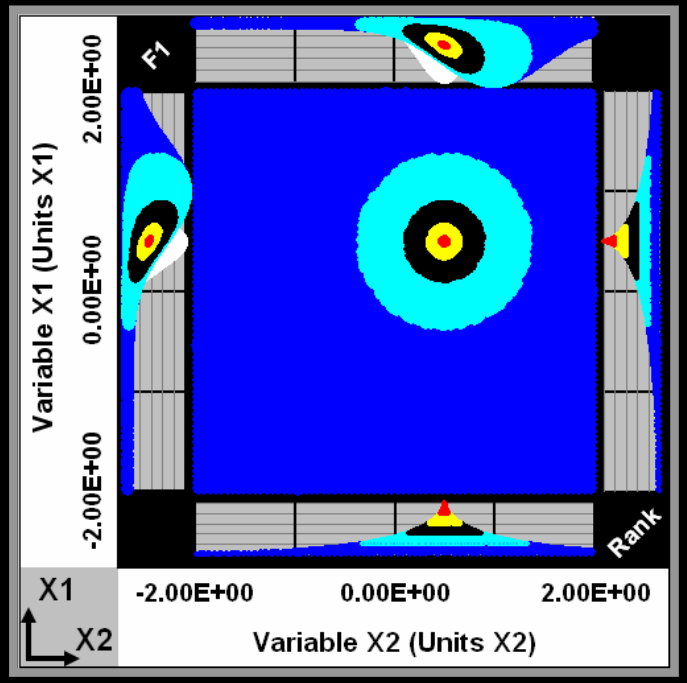
We find that the global solution lies at the origin, midway between the Maxima for F1 and F2.

2-D Trade-Off

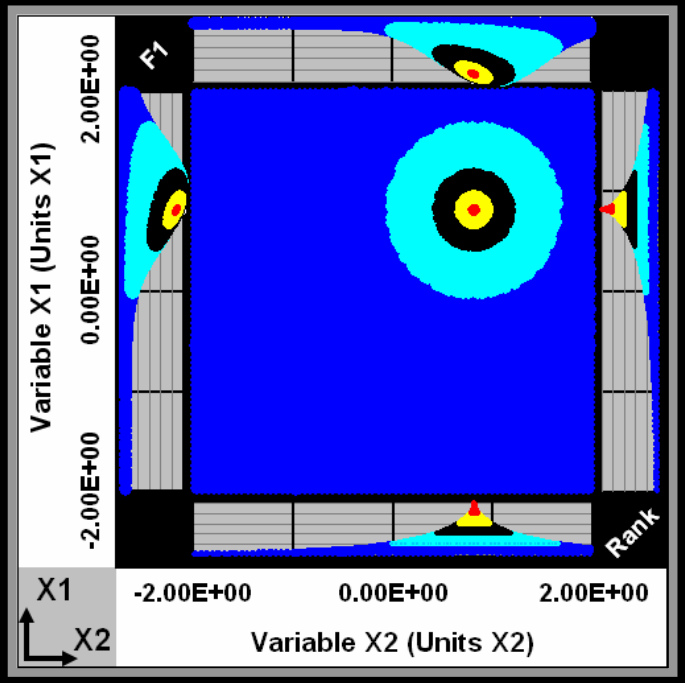
Since this is a Trade-Off rather than a Simultaneous solution, increasing the Weight (Wt) for F1 relative to F2 will shift the FOM to solutions that increasingly favor F1.



F1 Wt = 1
F2 Wt = 1



F1 Wt = 3
F2 Wt = 1



F1 Wt = 10
F2 Wt = 1

The Figure Of Merit and Normalized Errors

Regardless of whether we are seeking solutions which maximize or minimize a function, or which will set the function equal to a specific target value, every optimization problem can be mathematically formulated as a minimization problem. Without loss of generality, we will therefore confine the following discussion to minimizing a function:

$$F(X1, X2, \ldots Xn) = Min$$

within some specified n-dimensional search volume Vn. We will denote F_i as the value of F at any particular set of coordinates within Vn, Fmin as the minimum (best) value of F within Vn, and Fmax as the maximum (worst) value of F within Vn. The values of F_i can be rank ordered from best to worst using the **FOM** (Figure Of Merit) defined as:

$$FOM = (Fmax - F_i) / (Fmax - Fmin)$$

From which it follows that

$$0 \leq FOM \leq 1$$

and

FOM = 1 at the best solution
= 0 at the worst solution

The Figure Of Merit and Normalized Errors

The Normalized Function Error $\mathbf{dF_N}$ can then be expressed as

$$\mathbf{dF_N = 1 - FOM}$$

$$\mathbf{= (F_i - Fmin) / (Fmax - Fmin)}$$

Similarly, we will define the Normalized Error for each solution variable X as

$$\mathbf{dX_N = | (X_i - Xbest) / (Xworst - Xbest) |}$$

where Xbest is the value of X_i producing the best value of F found within Vn, and Xworst is the observed value of X_i which is the greatest distance away from Xbest. Like the **FOM**, the Normalized Errors $\mathbf{dF_N}$ and $\mathbf{dX_N}$ lie between 0 and 1.

Sensitivity Plots present the
Normalized X-error dX_N versus Normalized Function Error dF_N
on a logarithmic scale.

The Figure Of Merit and Normalized Errors

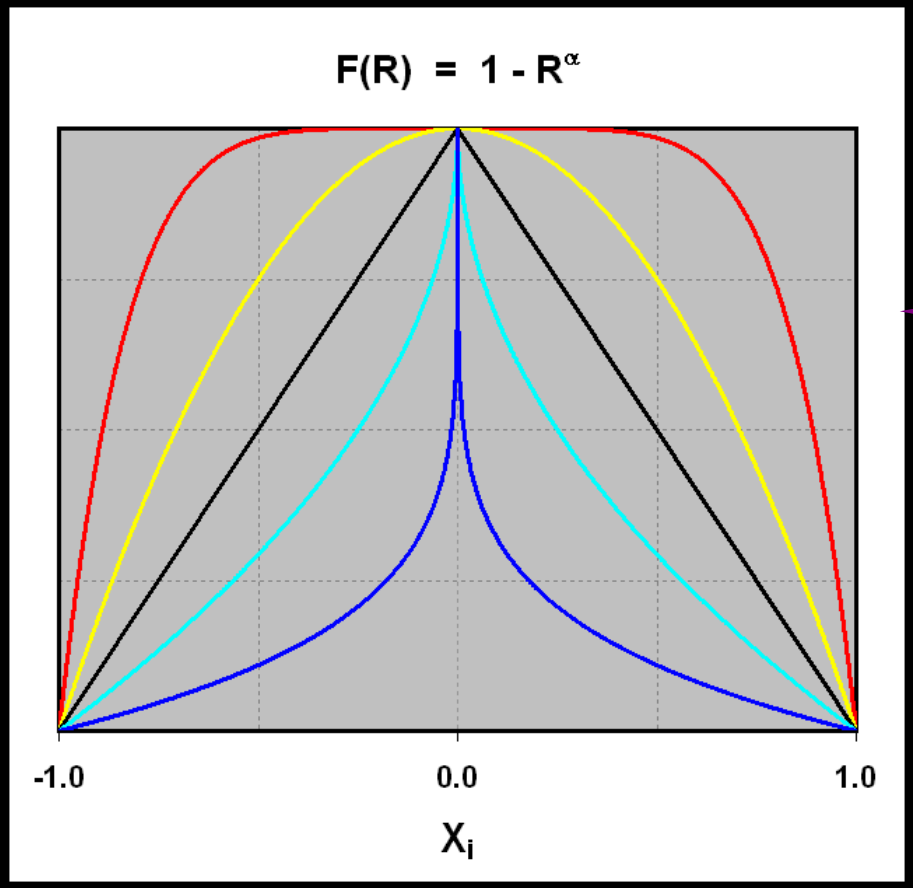
Multi-Objective Optimization

In multi-objective optimization, we seek to simultaneously optimize two or more functions. In this case, the Figure of Merit is defined as the weighted product of the individual **FOM**$_i$ values for each of the functions (i = 1 to m) to be optimized:

$$\mathbf{FOM} = \prod_{i=1}^{m} (\mathbf{FOM}_i)^{W_i}$$

where $\mathbf{W}_i$ is the weight (Wt) value specified by the user for each activated function to be optimized. Increasing the weight value $\mathbf{W}_i$ specified for an activated function will increase the sensitivity of the **FOM** to that function, thereby increasing the relative importance of that function in determining the best solution. If a weight value of zero is specified for an activated function, then that function will play no role at all in determining the best solution.

PSE Algorithm Convergence Rates



How rapid is the "Rapid" Search Mode?

For an extremely broad range of problems with unique solutions such as maximizing any of the family of radially symmetric functions shown here,

the convergence properties of the PSE algorithm are easily summarized. The "Volumetric Rate" V_{RATE} (that is, the factor by which the Error Volume of the X's is reduced on average for each iteration) is typically

$$V_{RATE} \approx 0.25$$

and generally lies within the range 0.15 to 0.30, regardless of the number of solution variables N_x

This means that after only 10 iterations of the PSE algorithm, the initial search volume specified by the user will be reduced by $(.25)^{10}$ which is more than a factor of One Million.

The minimum number of function evaluations required in each iteration is the Pod size defined on slide <24>. If we have a function of $N_x = 50$ variables, for example, the Pod size is 50 and the X-error volume is reduced on average by 75% for every 50 function evaluations!

PSE Algorithm Convergence Rates

Example: Minimizing the Quadratic Bowl

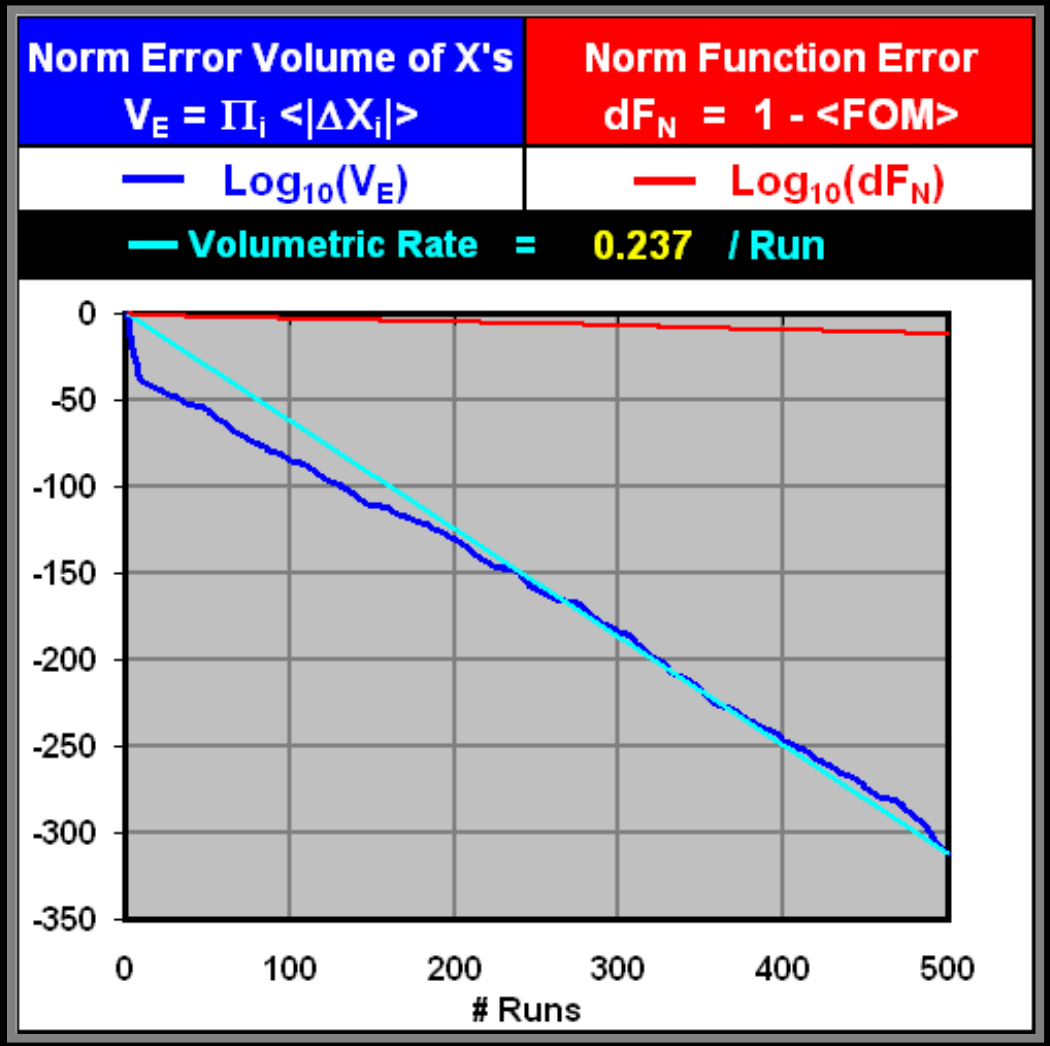
$$F(X_1, X_2, ..., X_n) = \sum_{i=1}^{n} X_i^2$$

with n = 50 variables

As shown on this graph,
the Error Volume is reduced by more
than 300 orders of magnitude

$$(0.237)^{500} = 2x10^{-313}$$

in 500 iterations.



Relating Solution Precision and Volumetric Rate

If we define the precision in each X solution as

xP = |Error in X| / |Initial Search Range for X|

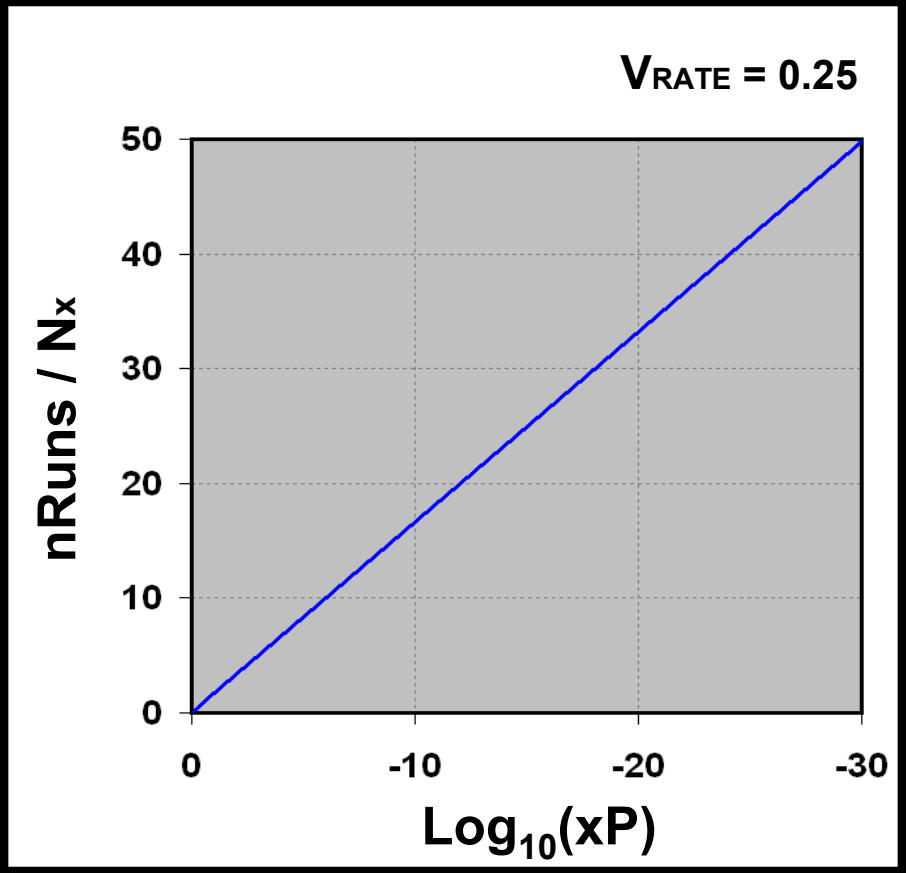
and we define nRuns as the number of iterations required to obtain that degree of precision for each X solution, then:

$$(V_{RATE})^{nRuns} = (xP)^{Nx}$$

The required number of iterations therefore varies linearly with the number of solution variables Nx:

$$nRuns = N_x \{ \text{Log}(xP) / \text{Log}(V_{RATE}) \}$$

The results for a Volumetric Rate = 0.25 are shown here.



For Nx ≥ 10, there are Nx function evaluations in each iteration, and thus the required number of function evaluations scales in proportion to $(N_x)^2$.

Convergence Rate for Multiple Solutions

In cases where the solution is not unique and we have many local optima, using a sufficiently large population of particles [Np > (# solutions)*(Pod size)] will permit simultaneous convergence at each solution with the same Volumetric Rate as in the case of a unique solution.

For more information, or to obtain an evaluation copy of the Andromeda software, contact:

Jim.Durnin@charter.net

Optimal Results
15822 Freedom Lane
Apple Valley, MN 55124
(952)953-3765