KALLISTOS

Version 1.5

34 Leopold St
Oxford, OX4 1TW
United Kingdom
Phone + 44 1865 324998
Fax + 44 1865 324991
info@lambda-tek.com
www.lambda-tek.com

Lambda Tek
34 Leopold St
Oxford, OX4 1TW
United Kingdom

Phone +44 1865 324 998
Fax +44 1865 324 991
Email: info@lambda-tek.com

CONTENTS

Chapter 1 Introduction

1.1 What is KALLISTOS

KALLISTOS is a novel optimisation tool capable of automatically improving existing designs of photonic devices with minimum intervention from the user. Using KALLISTOS, design cycle times of a new device can be dramatically reduced.

Considerable work has gone into all aspects of the product, resulting in a tool incorporating state of the art optimisation algorithms, combined with a powerful graphical user interface making it easy for the user to set up and run a design optimisation calculation, and a comfortable visual interface which allows the user to fully explore the solutions found.

Under the hood, KALLISTOS is equipped with several robust optimisers for the global and local optimisation of continuous functions. Moreover much work has gone into KALLISTOS to make it as efficient as possible. For example, in calculating the sensitivities used by the local optimiser, KALLISTOS takes full advantage of the special structure of the wave equations describing light propagation through photonic devices.

1.2 Installation Notes

[Kallistos stand-alone license only]

If you have a Kallistos stand-alone license and one or more of your products are under maintenance, you should be careful when receiving updates.

If an upgrade for one or more of the Photon Design main product(s) is received, the user must be aware that the Kallistos stand-alone license must also be upgraded. The Kallistos stand-alone plugin file must use the same major release of the shared file pdCore(x).dll.

The user may experience problems if some Photon Design products are not maintained. It is recommended that the user maintain all Photon Design products to ensure trouble-free operation.

1.3 Changes Log

New for Version 1.4

- Now supports *Named Variables*. The device to be optimised is now parameterised by setting the device parameters equal to an expression depending on some *named variables*. The optimiser will vary any *Named Variable* declared in the list of *Independent Variables* in the optimiser. This simplified system eliminates the use of *dependent variables*, which have therefore been removed.

- The concept of a work node is now obsolete by using the variables. The changes are automatically reflected to the source node. You can return to the original device simply by changing the variables.

New for Version 1.1:

- Added **minboxsize** variable for global optimiser.
- Introduced command line API
- Improved select columns dialog and select graphs dialog
- Greatly enhanced syntax in *dependent variables* and *user-defined objective function* definitions: Can now use array syntax for defining sequences of similar commands or instructions. Also introduced "sum" and "prod" routines in function parser. Can also use independent variables in objective function definitions.
- Enhanced *hypercube view.*
- Can now use independent variable in *user-defined objective function*

1.4 How it works

In the first iteration of a product design, the user typically starts with an initial guess of a design. The selected *parameters* and *constraints* defining the are then declared within KALLISTOS, as well as the quantity that needs to be optimised (the *objective function*) e.g. Power Transmission. Finally the calculation is launched with one of the optimisation algorithms available in KALLISTOS.

1.5 The function parser

Although KALLISTOS comes with a variety of predefined *objective functions*, KALLISTOS also comes with a powerful built-in function parser making it possible to define arbitrarily complex parameter constraints and *objective functions.*

1.6 The optimisation algorithms

Finding solutions to complex optimisation problems can be a difficult task. In general there is no optimisation algorithm that works in all cases. KALLISTOS comes with efficient algorithms based on recent advances in mathematical optimisation, each with complementing strengths:

- An efficient local descent routine, ideal for large, computationally intensive structures. An optimum can be found with a relatively small number of iterations.
- A deterministic global optimisation technique. This converges more slowly than the above, but is more likely, if not guaranteed, to find a global optimum.

1.7 The monitoring interface

Far from adopting a black box approach, the software comes with a powerful graphical monitoring interface for following the progress of multidimensional optimisation calculations. These prove to be particularly useful for detecting potentially optimal designs when using a global optimiser. For example, the user can quickly locate a point of interest in the parameter space and "home in" on it with a local optimiser.

1.8 Highly speed optimised

[FIMMPROP only]

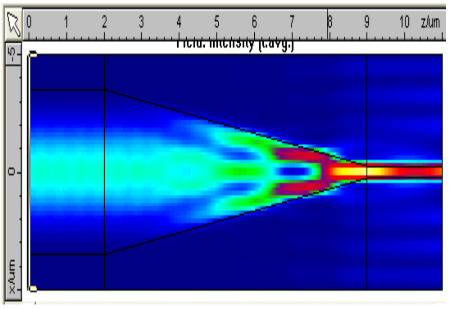
Optimisation calculations can be very lengthy for complex, 3D structures. We have exploited to the full, the mathematical structure of the wave equations, for example, by incorporating an analytical procedure for calculation of the sensitivities, essential to the optimisation process. This and other improvements make KALLISTOS an extremely fast optimisation tool for the design of photonic devices.

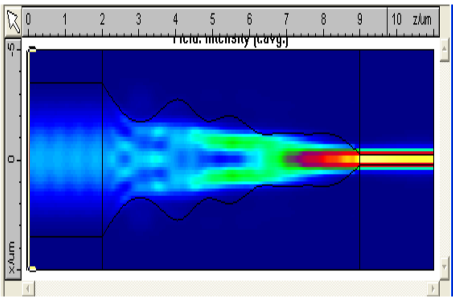
1.9 Learning KALLISTOS

Application specific tutorials are provided as separate documents. These can be found in the Documents section of your CD Image.

Kallistos Tutorial for Fimmwave:

- Using KALLISTOS to optimise a taper shape.

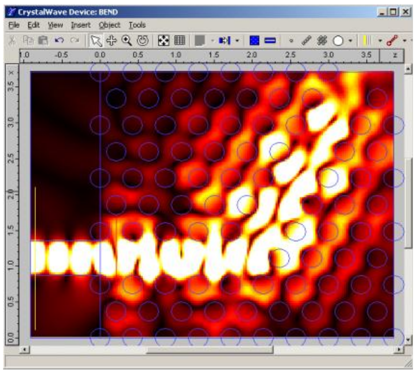


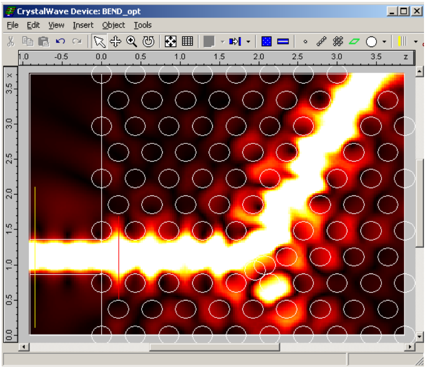


Left: Transmission 53% **Right: Transmission 91%**

Kallistos Tutorial for CrystalWave:

- Using KALLISTOS to minimise the reflection around a 60-degree bend in a photonic crystal slab.




CrystalWave Device: BEND_opt
File Edit View Insert Object Tools

Left: Reflection 38% **Right: Reflection <1%**

Kallistos Tutorial for OmniSim.

- Using KALLISTOS to optimise a ring resonator structure.

Chapter 2

The KALLISTOS Framework

2.1 Using KALLISTOS

Once an optimiser has been added to a project the following optimisation settings need to be configured:

- The device which you wish to optimise.
- The specific parameters of this device that you want to optimise
- The technique you wish to use to optimise the function.
- The objective function, which will numerically indicate the how good the device is performing.
- The finishing conditions which will determine that the optimisation is complete.

Once you have done this and are satisfied that all the settings are correct you can run the optimisation. You can either continue from where a previous run left off or start an optimisation from the very beginning.

There are various ways in which you can analyse the optimisation process as it proceeds or once it has finished.

- Using the graphical views
- Selecting points and looking at the function value

2.2 Setting up the optimisation

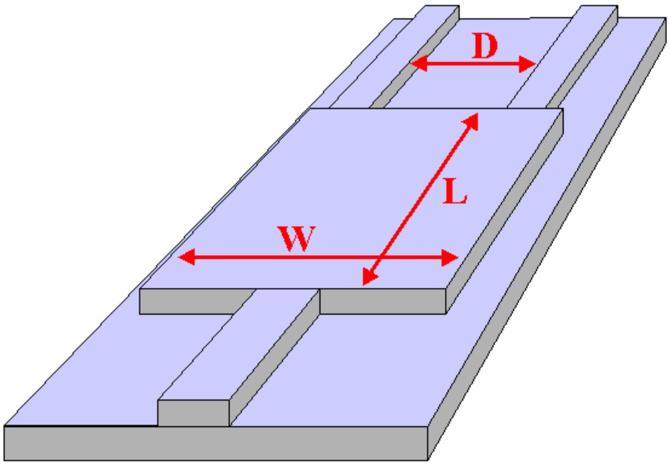
2.2.1 Selecting the Device to be Optimised

To set the device to be optimised click on on the tool bar, or via the menu **/Source node/Set Source node**. This is referred to as the *source node*, you can then choose a node from the *node selector*. The *source node* can be any *node type* available in the application.

You can view the Source node you have just selected by pressing on the button, or via the menu **/Source node/View Source node**

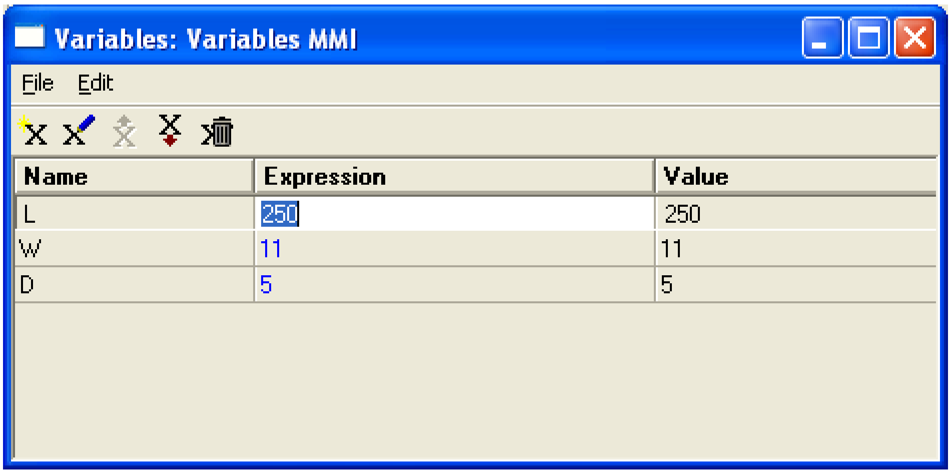
2.2.2 Parameterising the Device to be Optimised

At this stage you will have to identify which parameters of the device you want to optimise. For example, if the problem is maximising the power in the fundamental modes of the two output waveguides of an MMI coupler, one might choose to optimise the following parameters:



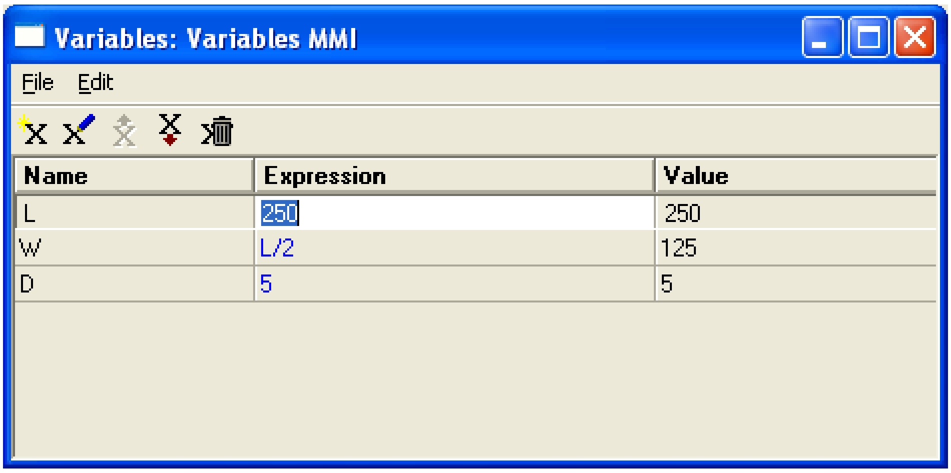
Obviously there are more parameters whose effect could be explored, refractive indices or etch depths to name a few, but it is advisable to reduce their number to a carefully chosen minimum in order to minimise execution time.

After identifying what parameters are to be optimised, the software needs to be told which they are and how they affect the device. This is easily implemented with the *Named Variables* system. In the above example, you could just add Variables node, with three variables in it:



Name	Expression	Value
L	250	250
W	11	11
D	5	5

and set the corresponding properties of the device to the values of these variables. Or you might want to specify a relation between L and W, say W=L/2.



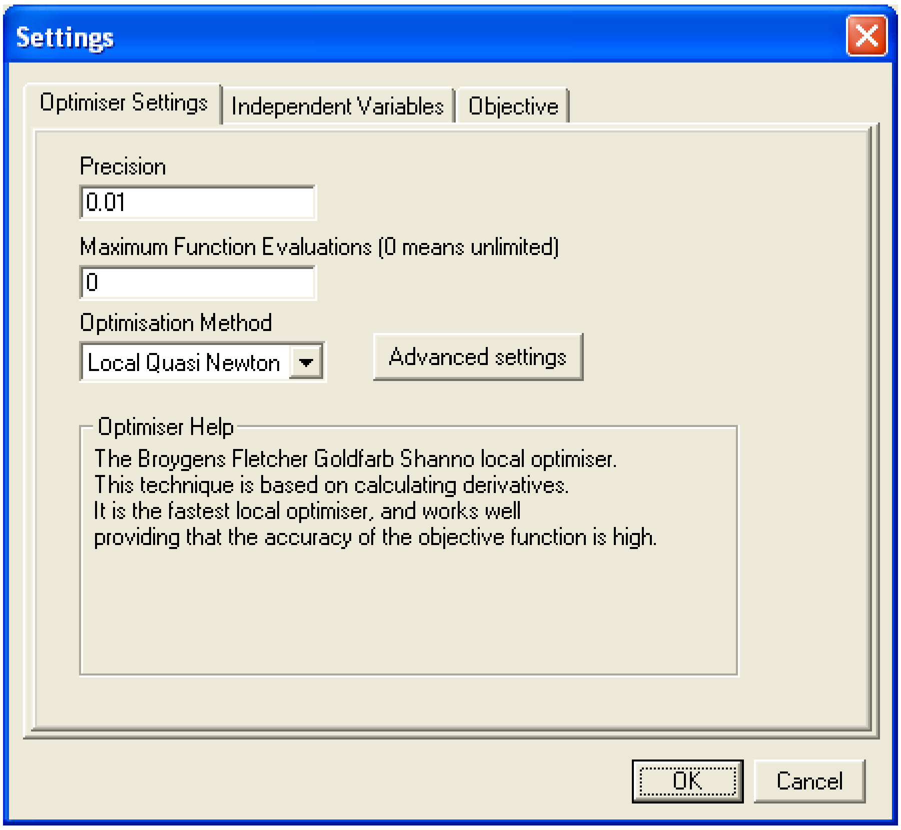
Name	Expression	Value
L	250	250
W	L/2	125
D	5	5

Generally, a property of a device can be set to a *Variable* or an *Expression*. See the chapter on Variables and Expression son your products manual for more information on that.

It is important however, that all the parameters you want to optimise are associated with a distinct variable. The reason is that KALLISTOS will vary directly the above variables when trying to optimise the structure. So in the above example, when you specify the independent variables, you should specify L,W and D if you want them to vary independently, but if you want W=L/2, the idependent variables should be only L and D. (see section 2.2.5 for describibg the *Independent Variables*)

2.2.3 Configuring the Optimiser

You can configure the optimiser by pressing , or via the menu **/Optimiser/Settings**. This will show the following screen



Precision
0.01

Maximum Function Evaluations (0 means unlimited)
0

Optimisation Method
Local Quasi Newton

Optimiser Help
The Broygens Fletcher Goldfarb Shanno local optimiser.
This technique is based on calculating derivatives.
It is the fastest local optimiser, and works well
providing that the accuracy of the objective function is high.

As you can see, there are three tabs.

- The *Optimiser Settings* tab, where you select the optimization method to use and the precision of your optimiser,
- *the Independent Variables* tab, where you specify the variables that the optimiser will try to optimise, and the
- *Objective* tab, where you have to specify your objective function, in other words, to give the optimiser a criterion to decide if a specific structure is good or bad.

2.2.4 The Optimisation Methods

The Help box is there to explain a little about each optimiser and when it should be used. Currently the following optimisation methods are available. See §2.5 for more details on these algorithms.

Null This optimiser makes no changes in the optimisation process (hence its name). It evaluates the *objective function* at the starting point specified in the parameter settings and does no optimisation. Its main purpose is to test the parameter setup of the problem.

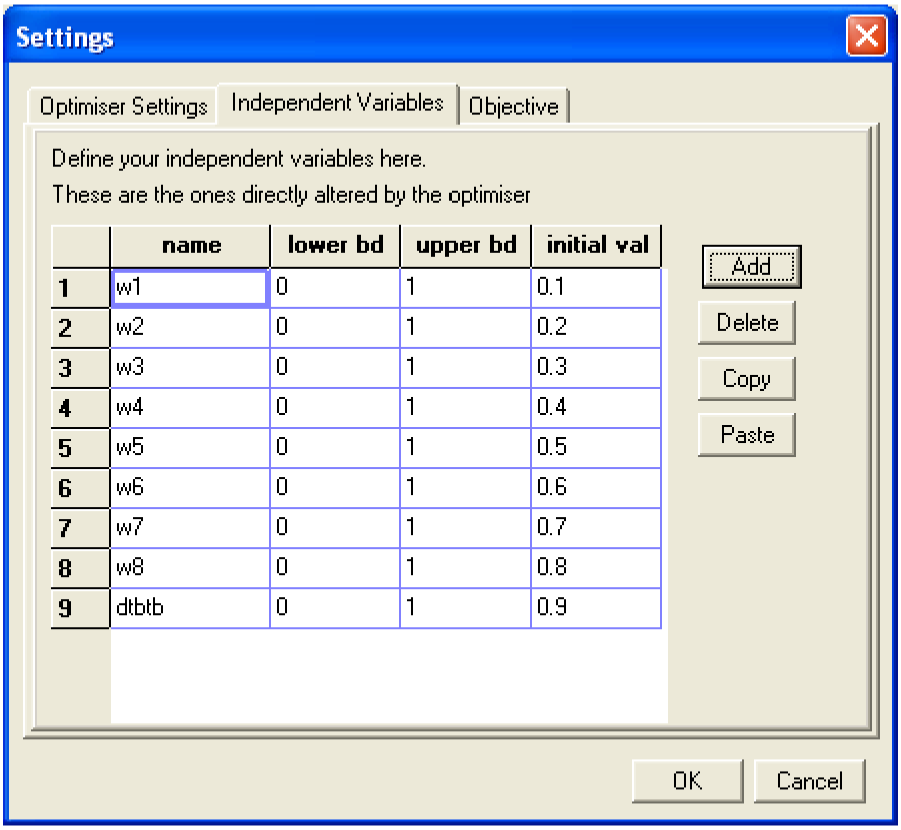
Local Quasi Newton This optimiser uses derivatives to converge on the nearest local optimum. The algorithm will stop when the **Maximum Function Evaluations** is reached or when the improvement in the objective between two successive iterations is below the **precision**. The derivatives can be calculated in various ways. See §2.5.3 for details on this topic.

Global This optimiser uses a deterministic algorithm, meaning that if you run it long enough, it is **eventually** guaranteed to find an optimum. As with the *Local Quasi Newton* method, the algorithm will stop when the **Maximum Function Evaluations** is reached or the improvements in the objective between successive iteration is below the **precision**. Note however that the precision will only control how far the solution is from the best optimum found so far, which may not be the global optimum. A further setting is available in the advanced settings: **minboxsize**. See section 2.5.2 for more details.

The use of one of the above methods is recommended. There are two other optimisation algorithms included (*Local Powell* and *Local Simplex*), but they are still in experimental stage.

2.2.5 The Independent Variables

The *Independent Variable*s are the variables directly varied by the optimiser. The way the device changes when an *Independent Variable* is altered can be easily defined by the user through the named *Variable* system and the *Expression Evaluator*. See section 2.2.2 for a discussion on parameterising a device.



Define your independent variables here.
These are the ones directly altered by the optimiser

	name	lower bd	upper bd	initial val
1	w1	0	1	0.1
2	w2	0	1	0.2
3	w3	0	1	0.3
4	w4	0	1	0.4
5	w5	0	1	0.5
6	w6	0	1	0.6
7	w7	0	1	0.7
8	w8	0	1	0.8
9	dtbtb	0	1	0.9

The *Independent Variables* table contains four columns:

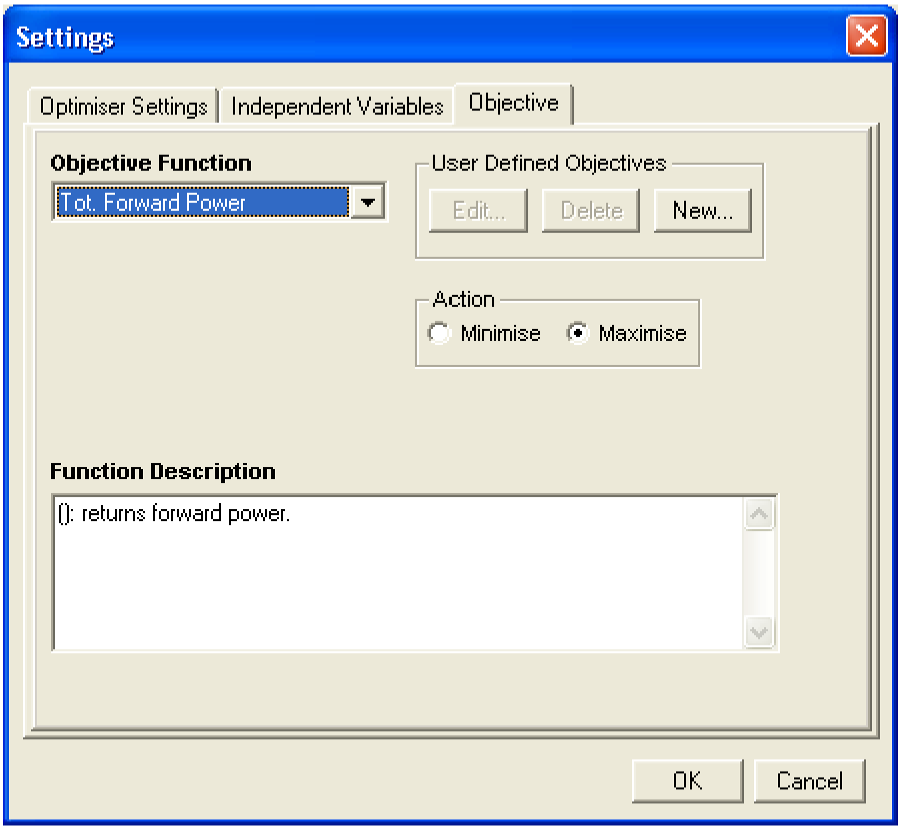
- **Name**. The user name given to an *Independent Variable*.

- **Min Value**. This is the minimum value that an *Independent Variable* can take. This will be the lower bound in the optimisation process.
- **Max Value**. This is the maximum value that an *Independent Variable* can take. This will be the upper bound in the optimisation process.
- **Initial value**. This is the starting/original value for an *Independent Variable*. The local optimisation method will use this initial value as a starting point for the optimisation. The initial value must of course be between the minimum and the maximum values.

Pressing **Add** will add a new row to the table corresponding to a new parameter. The parameter name, minimum, maximum and initial values need to be entered for every parameter. You can use the **Add**, **Delete**, **Copy** and **Paste** buttons to set up or modify the table. Copy and Paste can be used effectively if there are a lot of parameters required for the optimisation.

2.2.6 The Predefined Objective Functions

A list of *predefined objective functions* is available; these vary depending on the application. Therefore you will have to specify a source node in order for the list of predefined objective functions to appear. A brief description is given in the Function Description box on the Objective panel.



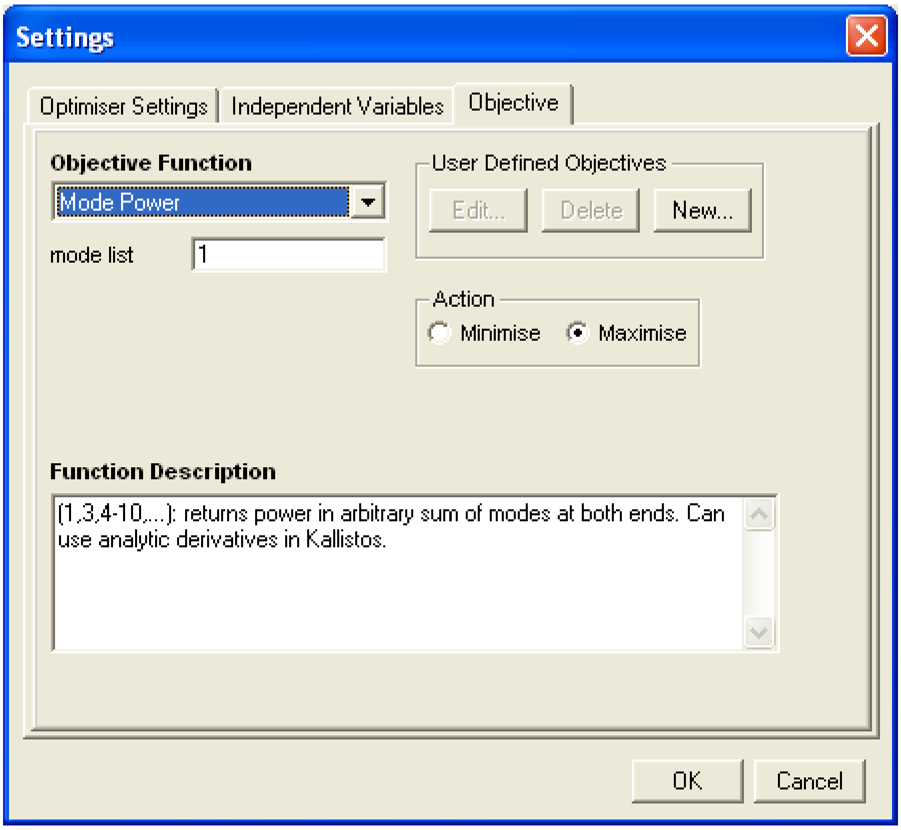
Objective Function: Tot. Forward Power

Action: ○ Minimise ● Maximise

Function Description: (): returns forward power.

You can choose to Maximise or Minimise the *objective function.*

Some predefined objective function take arguments. For example, if "Mode Power" is selected, a text box appears, titled "mode list". There you can specify the mode or modes for which the function should calculate the power.



Settings

Optimiser Settings | Independent Variables | Objective

Objective Function

Mode Power

mode list 1

User Defined Objectives

Edit... Delete New...

Action

○ Minimise ● Maximise

Function Description

(1,3,4-10,...): returns power in arbitrary sum of modes at both ends. Can use analytic derivatives in Kallistos.

OK Cancel

2.2.7 The User Defined Objective Functions

If you want to use an objective function other than the *predefined objective functions* then you can use the **New...**, **Edit...** and **Delete...** buttons.

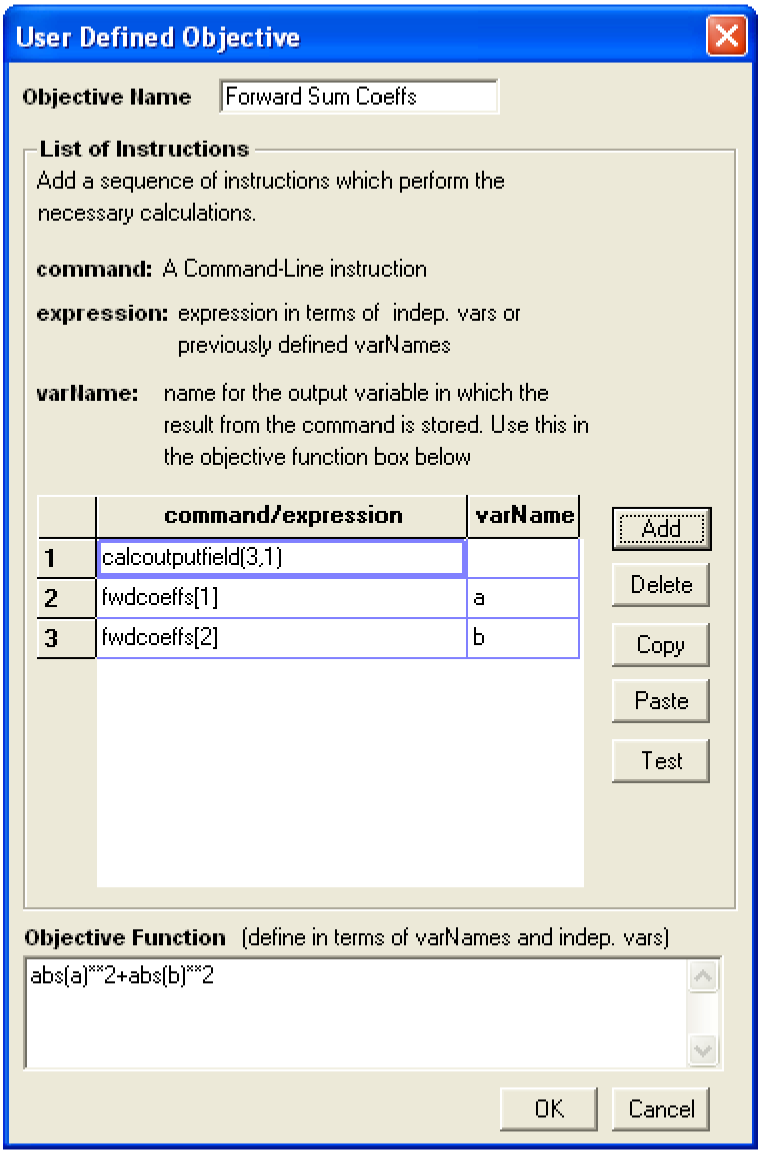
New.... allows you to create a brand new *user defined objective function.*

Edit... allows you to edit an existing *user defined objective function.* It is active only when a n existing user-defined objective function is selected.

Delete... allows you to delete an existing *user defined objective function* that you no longer want. It is active only when a n existing user-defined objective function is selected.

Defining a new Objective Function

The User Defined Objective Dialog allows you to create your own customised objective function by specifying a sequence of *Command-Line* instructions that KALLISTOS needs to perform in order to calculate the value of the *objective function.*



User Defined Objective

Objective Name: Forward Sum Coeffs

List of Instructions

Add a sequence of instructions which perform the necessary calculations.

command: A Command-Line instruction

expression: expression in terms of indep. vars or previously defined varNames

varName: name for the output variable in which the result from the command is stored. Use this in the objective function box below

	command/expression	varName
1	calcoutputfield(3,1)	
2	fwdcoeffs[1]	a
3	fwdcoeffs[2]	b

Objective Function (define in terms of varNames and indep. vars)

```
abs(a)**2+abs(b)**2
```

The Objective Function Dialog

Each line specifies a **command** - a *Command-Line* instruction to be executed by the application scripting engine, or an **expression** – an algebraic expression involving independent variables or *output variables* defined in previous lines. *Output variables* are optionally specified in the **varName** column, and are where the result of the *Command-Line* instruction or expression is stored. These can then be used in all subsequent expressions as well as the **objective function**.

The **Objective Function** must be a valid algebraic expression in terms of the *output variables* and independent variables.

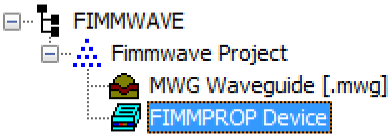
Each time KALLISTOS needs to calculate the objective, it will execute the list of instructions/expressions in the order that they appear in the list, and then uses the in built *function parser* to evaluate the *Objective Function*. See $2.2.8 for details on the *function parser*.

Writing and testing commands

In writing he command string you can press the <TAB> key to invoke the *command completion box* to help you find the appropriate object members.

Note that in the command line, each command will start with a root that corresponds to the location of the object to which it is applied.

For instance, if you calculate the scattering matrix of a FIMMPROP Device, which is located in the project tree as shown below



the command will read

```
app.subnodes[1].subnodes[2].update
```

where `app.subnodes[1].subnodes[2]` gives the location of the FIMMPROP Device in the project tree and `update` is the actual command for the FIMMPROP Device.

Whenever writing commands into KALLISTOS, you only need to write the last part of the command; the root of the device to be optimised will be added as a prefix.

So in this case if the FIMMPROP Device has been selected for the optimisation and you want to calculate the scattering matrix, you will only need to write

```
update
```

which Kallistos will send to FIMMWAVE as

```
app.subnodes[1].subnodes[2].update
```

Once you have completed writing the list of commands and expressions, you can test it by pressing **Test**.

Basing new Objective Function on a existing one

When you press **New...**, you will be asked if you want to copy the currently selected objective function. If you press **yes** then the commands of the currently selected objective function will be copied into the new objective function. You can use this as a convenient template for building your customised objective functions.

Writing sequences of Instructions

It is possible to write a sequence of similar instructions using convenient shorthand. You do this by beginning the **command/expression** box by

```
[I=a:b]
```

where I is the *index variable*, which can be any character or sequence of characters, and a,b two numbers with $b \geq a$. The index variable can then be used in both the **command** or **function** columns.

For example, supposing you have the following lines in the dependent variables table:

fwdcoeffs[1]	a1
fwdcoeffs[3]	a2
fwdcoeffs[5]	a3
fwdcoeffs[7]	a4

Then you can write these as one line:

[I=1:4] fwdcoeffs[2*I-1]	a

where “a” now is a **vector** variable with 4 elements a[1] a[2] a[3] a[4].

These can then be conveniently used in subsequent **expressions** or the **objective function** using the “sum” or “prod” functions. For example an objective of the form

```
abs(a1)**2+ abs(a2)**2+ abs(a3)**2+ abs(a4)**2
```

can now be written as:

```
sum(I=1:4,abs(a[I])**2)
```

2.2.8 The function parser

KALLISTOS comes with an in-built *function parser* used for evaluating expressions defined in the optimiser settings.

The form of these expressions follows a FORTRAN type syntax and if used to describe a material in a materials database file should be enclosed in double quotes.

The usual basic operations are available

“+”, ”-“, ”/” , ”*” ,”**” ,

functions:

“sin”, “cos”, “tan”, “exp”,
“asin”, “acos”, “atan”, “ln”,
“sinh”, “cosh”, “tanh”

and the predefined fundamental constant _PI that stands for “π”.

In addition, there are the following functions and constants:

sqrt(a)	returns the square root of “a”
sign(a)	returns +1 if a>0, 0 if a=0,-1 if a<0
pow(a,b)	same as a**b, i.e. returns a to the power of b
abs (a)	(a real or complex) returns |a|
arg(a)	(a complex) returns the argument of a
spline(x,a1,b1,a2,b2,…)	returns value at x of spline running through list of points (a1,b1),a2,b2),…
lininterp(x,a1,b1,a2,b2,…)	returns value at x of piecewise linear interpolation running through list of points (a1,b1),a2,b2),…
max(a1,a2,…)	returns maximum value in list a1,a2,…
min(a1,a2,…)	returns minimum value in list a1,a2,…

“sum” and “prod” functions

The function parser also has a special syntax for performing sums and products. These are of the form

```
sum(I=exp1:exp2,exp3)  or  prod(I=exp1:exp2,exp3)
```

where:

`exp1,exp2` can be any number or expression in terms of defined parameters

`exp3` is an expression in terms of defined parameters, and the *variable index* I.

If *exp2>=exp1*, `exp3` will be summed/multiplied in I **increasing** in unit steps from `exp1` to `exp2`. Otherwise, `exp3` will be summed/multiplied in I **decreasing** in unit steps from `exp1` to `exp2`.

2.2.9 Advanced editing

In optimisation problems involving so many variables, or complex expressions, it may prove to be more convenient to edit the optimiser settings directly via the *text block* representation. Bring up the *text block* by clicking on abc 014 or selecting **/Optimiser/Edit settings as text.**

The *text block* syntax is fairly self-explanatory. Rather than explaining all the keywords and format here, we suggest doing an initial set-up via the settings dialog first, and only then adding more settings directly via the text block.

2.3 Running the optimisation

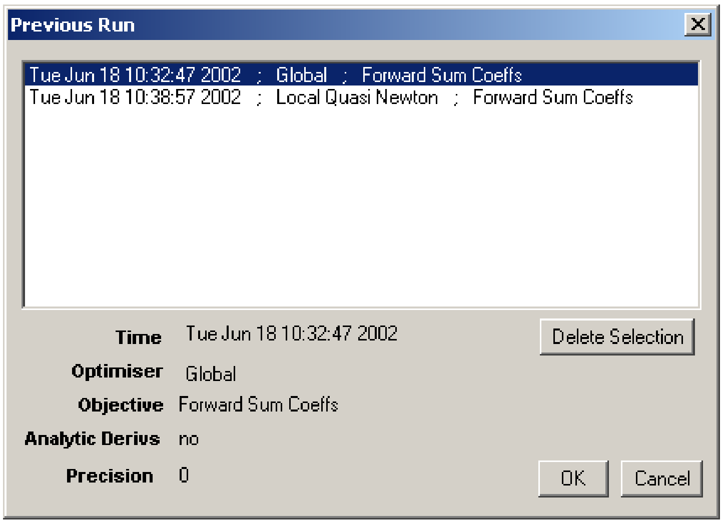
Pressing the Run tool ⚡ will start the optimisation process. This will continue until the finishing conditions have been satisfied or the user kills the run via their application.

2.3.1 The caching mechanism

As the calculation proceeds, all the calculated data points are stored to disk. This allows the user to resume a previous optimisation run. If you have chosen the wrong finishing conditions and the run has finished too early then you can change the finishing conditions and press run again to continue from where you left off.

Selecting a cache

You may select a previous run by clicking on 📂 or selecting **Optimiser/Previous Calc.** The dialog that appears will give you a list of the previous runs that you can select, with the date on which the cache was recorded, the optimisation method used and the precision of the optimisation. On closing the dialog, the optimiser settings will be updated to the ones in the selected cache, as well as all the views. Note that on opening the dialog, the last cached run is highlighted.



```
Tue Jun 18 10:32:47 2002  ;  Global  ;  Forward Sum Coeffs
Tue Jun 18 10:38:57 2002  ;  Local Quasi Newton  ;  Forward Sum Coeffs
```

Time Tue Jun 18 10:32:47 2002

Optimiser Global

Objective Forward Sum Coeffs

Analytic Derivs no

Precision 0

Selecting a cache

2.3.2 Resetting an optimisation

KALLISTOS automatically resumes an interrupted optimisation run. You may start an optimisation run from scratch by clicking on ↶ or selecting **Optimiser/Reset Calc.** Note this will delete the cache file containing the current run.

2.3.3 Recovering an optimisation run

KALLISTOS comes with a recovery mechanism in case the program unpredictably shuts down during an optimisation run. This is an event that hopefully will not arise, but it could be very frustrating if KALLISTOS were left to run an optimisation overnight, only to find the program crashed sometime before arriving the next morning!

KALLISTOS can recover this run precisely because each run is saved to disk as it proceeds. In such an event you may recover the lost data **providing that the project was saved immediately before the optimisation was started**. In this case, opening the same project again will automatically reload the interrupted run from disc.

2.4 Analysing the optimisation

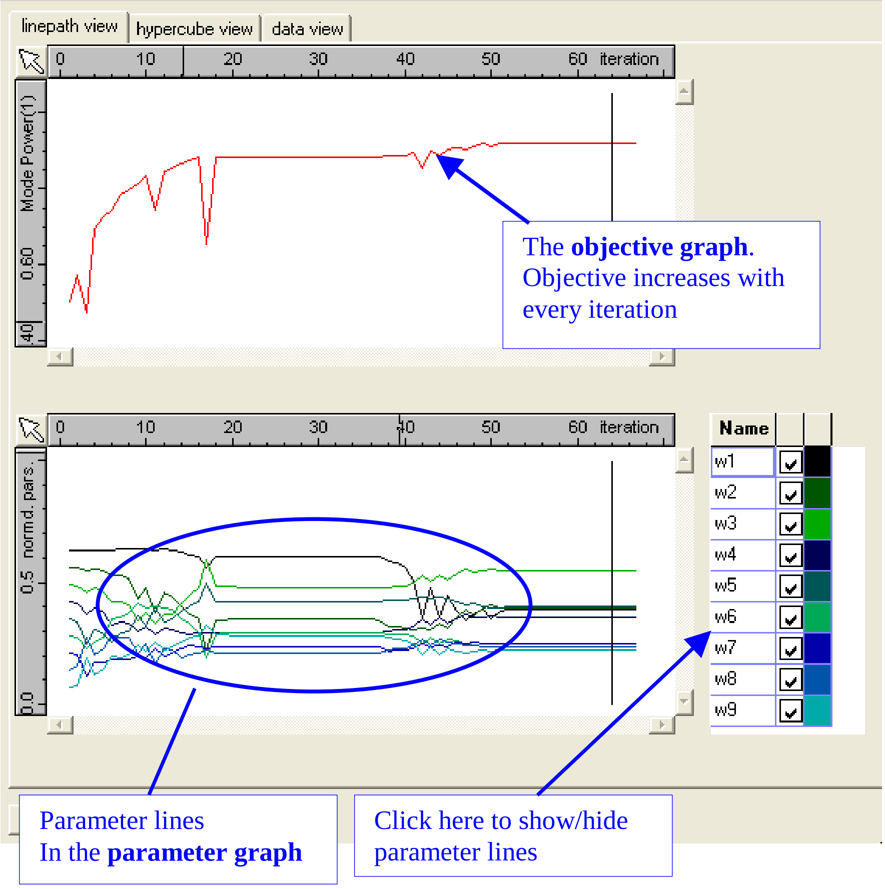
2.4.1 The Graphical Views

Different views are provided to allow the user to analyse the optimisation process that they have carried out.

Clicking on the **Detach View** button will detach the currently highlighted view in the notebook and place it in a separate window. Highlighting a point in one view will cause the same point to be highlighted in all other views. The point's details will also be displayed in the selected point table. Clicking on the different tabs of the notebook allows you to toggle between views.

Line Path View

Currently this is only used by the Local Quasi Newton optimiser.



Name		
w1	☑	■
w2	☑	■
w3	☑	■
w4	☑	■
w5	☑	■
w6	☑	■
w7	☑	■
w8	☑	■
w9	☑	■

The **objective graph**. Objective increases with every iteration

Parameter lines
In the **parameter graph**

Click here to show/hide parameter lines

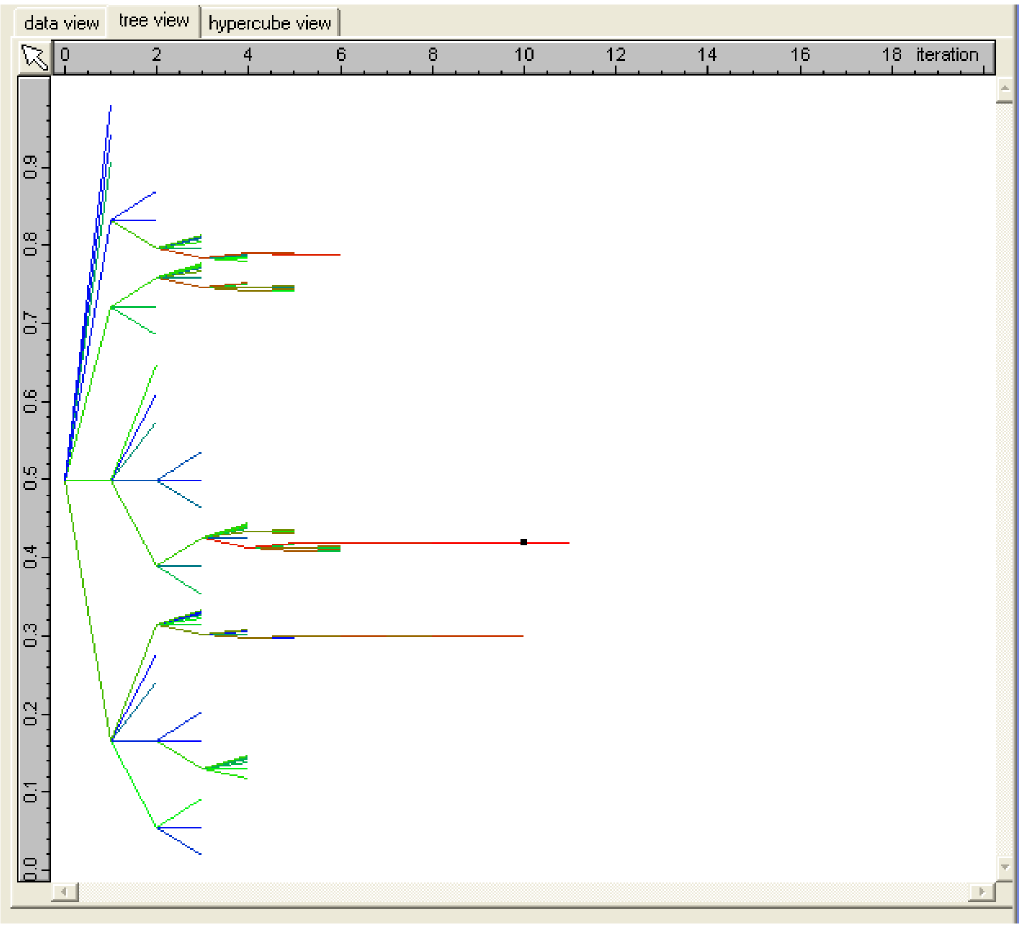
The top graph shows the *objective function* value for each iteration. If a user-defined objective function is being used, it will also show any *output variables* defined in the objective function instruction list. The bottom graph shows the corresponding parameter values for each iteration.

To highlight a point, click on the appropriate point on the line in either of the two graphs. A vertical black line at that iteration will mark that you have chosen to highlight that point.

The table on the right hand side of the parameter graph shows the colours used for each parameter and whether or not the parameter is being displayed. Check or uncheck a parameter to display or hide it.

Tree View

The *tree view* is one of the monitoring interfaces within KALLISTOS for viewing the progress of a global optimisation algorithm of any multidimensional problem. The longest branches tend to correspond to potentially good designs. Just click on any of these to view the corresponding design.

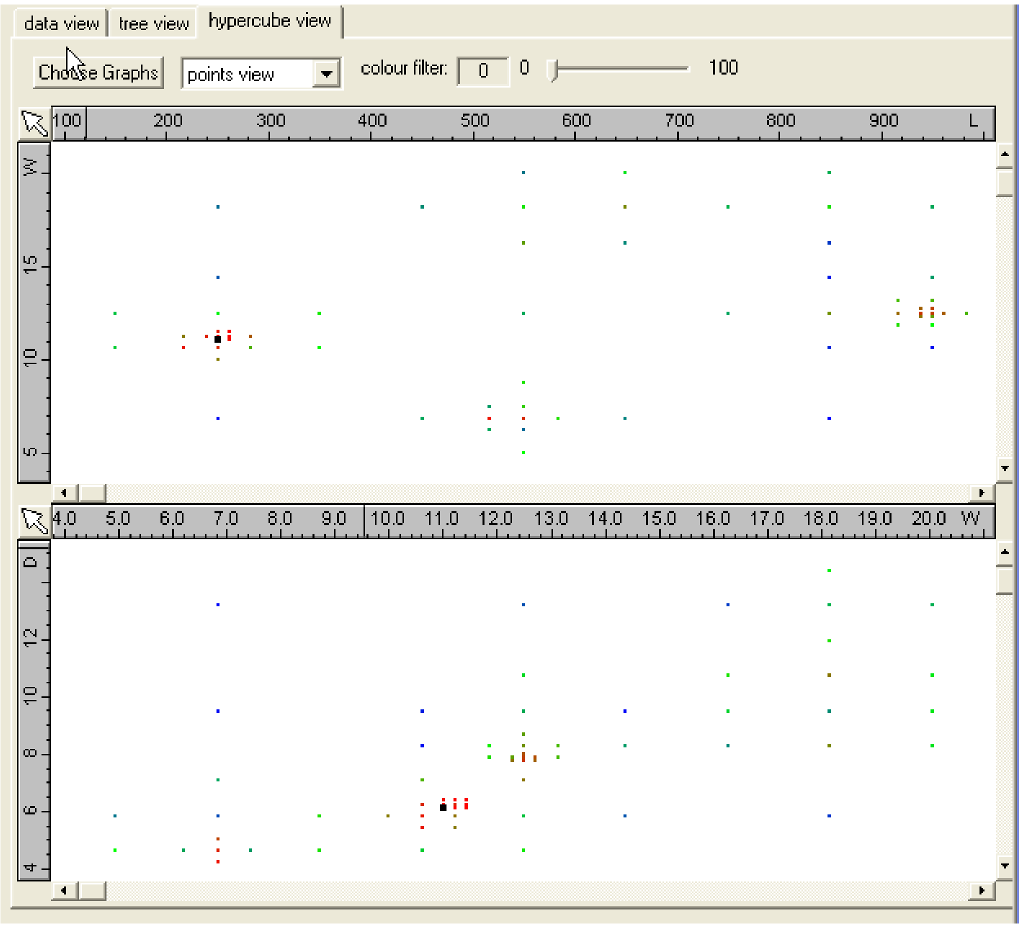


Each branch corresponds to a hyper-rectangle of the solution space that is being explored. The line colour indicates how good the function value is. It goes from blue for the worst function value, through green, to red for the best one. This is only used by the Global optimiser.

To highlight a point, click on the appropriate branch in the *tree view*. This point is highlighted on the tree view with a rectangle.

Hypercube View

Each graph is a 2D graph of one parameter against another. . The colour indicates how good the function value is. It goes from blue for the worst function value, through green, to red for the best one.



points view

colour filter: 0 0 100

To highlight a point click on the point any of the graphs and the nearest point will be highlighted on all graphs of the *hypercube view*.

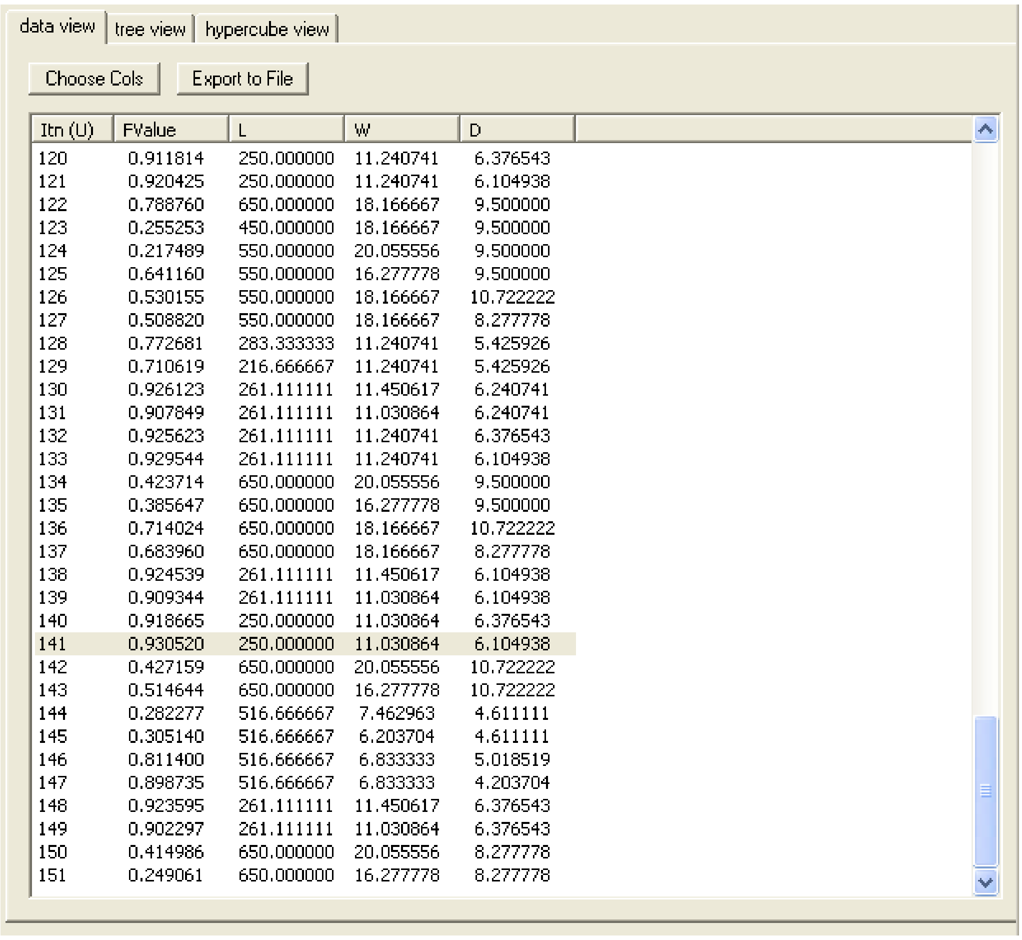
Apart from this default *points view* you can also choose the *RGB view* or the *shade view*. Both of these views display the calculated points as regions "filled out" to cover missing gaps, and therefore indicate more clearly the regions with better function values. You can use the **colour filter** to highlight regions with the better function values. Increasing the colour filter progressively will ignore any regions with non-optimal function values, and will eventually highlight only the best point.

The difference in the *RGB view* and the *shade view* is only in the colour scheme: the *RGB view* goes from blue, through green, to red for points with better function values, while the *shade view* goes from a dark to light colour.

Selecting Graphs – If there are many variables you wish only to view some of the possible graphs. You can use the **select all** and **select none** buttons to speed up the process of selecting the graphs that you need.

Data View

This displays the raw data in numerical form for each iteration in a table. For each point the user can choose to display the iteration number, the function value and the parameter values.



Itn (U)	FValue	L	W	D
120	0.911814	250.000000	11.240741	6.376543
121	0.920425	250.000000	11.240741	6.104938
122	0.788760	650.000000	18.166667	9.500000
123	0.255253	450.000000	18.166667	9.500000
124	0.217489	550.000000	20.055556	9.500000
125	0.641160	550.000000	16.277778	9.500000
126	0.530155	550.000000	18.166667	10.722222
127	0.508820	550.000000	18.166667	8.277778
128	0.772681	283.333333	11.240741	5.425926
129	0.710619	216.666667	11.240741	5.425926
130	0.926123	261.111111	11.450617	6.240741
131	0.907849	261.111111	11.030864	6.240741
132	0.925623	261.111111	11.240741	6.376543
133	0.929544	261.111111	11.240741	6.104938
134	0.423714	650.000000	20.055556	9.500000
135	0.385647	650.000000	16.277778	9.500000
136	0.714024	650.000000	18.166667	10.722222
137	0.683960	650.000000	18.166667	8.277778
138	0.924539	261.111111	11.450617	6.104938
139	0.909344	261.111111	11.030864	6.104938
140	0.918665	250.000000	11.030864	6.376543
141	0.930520	250.000000	11.030864	6.104938
142	0.427159	650.000000	20.055556	10.722222
143	0.514644	650.000000	16.277778	10.722222
144	0.282277	516.666667	7.462963	4.611111
145	0.305140	516.666667	6.203704	4.611111
146	0.811400	516.666667	6.833333	5.018519
147	0.898735	516.666667	6.833333	4.203704
148	0.923595	261.111111	11.450617	6.376543
149	0.902297	261.111111	11.030864	6.376543
150	0.414986	650.000000	20.055556	8.277778
151	0.249061	650.000000	16.277778	8.277778

To highlight a point click on the row number on the left you wish to highlight and the whole row will be highlighted in blue.

Selecting Columns – If you have many variable you may wish to decide which variables you wish to view. You can use **select all** and **select none** buttons to speed up the process of selecting the columns that you need.

Sorting by Variables – Clicking on the column heading for a variable will allow you to sort by that variable either in increasing order or decreasing order

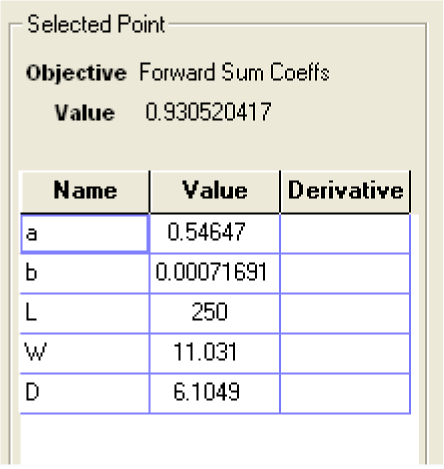
Export to File – You can export the contents of the data table to a text file. This can then easily be read into products like Microsoft Excel, Access or Word.

2.4.2 Selecting a Point

You can select a point of the optimisation, just by clicking on it, on any view. The data of that point (or iteration), will appear at the lower left of the KALLSTOS Optimiser window. The independent variables will be set to the values of that iteration and therefore the source node will be set to reflect those values.

Selected Point Info

This contains the parameter information for the selected point.



Selected Point

Objective Forward Sum Coeffs

Value 0.930520417

Name	Value	Derivative
a	0.54647	
b	0.00071691	
L	250	
W	11.031	
D	6.1049	

Value is the value of the objective function at the selected point

The **Name** column contains the variable name

The **Value** column contains the value of that variable at that point

The **Derivative** column contains the value of first derivative of the *objective function* with respect to the variable.

The derivative is another term for the differential. This can be viewed as the Sensitivity of the function with respect to changes in the variable.

Select Best Point

This selects the calculated point to be displayed with the best function value. This point will be highlighted on all views.

Update Initial

This allows you to update the initial value of the variables with those of the selected point

2.5 The Optimisation Algorithms

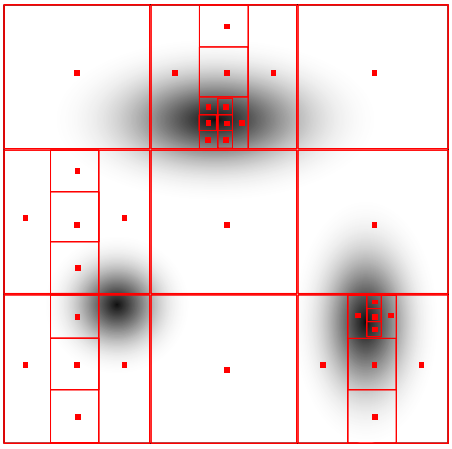
KALLISTOS currently comes with a Local and Global optimisation method. Here we give a description of how they work, plus some tips on how to use them. You may also notice that two other Local methods appear in the list, named *Local Powell* and *Local Simplex*, but their use is not recommended, as they are still in experimental stage.

2.5.1 The *Local Quasi Newton* method

This method uses derivatives to converge on the nearest local optimum by descending/ascending towards the lowest/highest function value depending on whether you are minimising or maximising the *objective function*. It is a globally convergent algorithm, which means that it will home in on the (local) optimum however far away it is. It is also has a second order of convergence rate when sufficiently close to the optimum. The method however requires the derivatives of the *objective function*. The way these are calculated depends on the *Advanced Settings* – see the next section for details. The algorithm will stop when the **Maximum Function Evaluations** is reached, or when the improvement in the objective between two successive iterations is below the **precision**.

2.5.2 The *Global* method

This is a deterministic algorithm, meaning that if you run it long enough, it is **eventually** guaranteed to find an optimum. This is in contrast with stochastic type algorithms (e.g. genetic, evolutionary, simulated annealing strategies) which may at times prove quicker, but are by no means guaranteed to find the global optimum. As illustrated below, this algorithm follows a systematic splitting strategy: It divides all the areas of the parameter space, but sub divisions happen more frequently in those areas deemed more likely to have an optimum. Eventually all points in the parameter space are covered, which is why it is deterministic.



The problem with all global optimisation routines is that it is not generally possible to predict just how many iterations it will take. As with the *Local Quasi Newton* method, the algorithm will stop when the **Maximum Function Evaluations** is reached or the improvements in the objective between successive iteration is below the **precision**. Note however that the precision will only control how far the solution is from the beast optimum found so far, which may or may not be the global optimum.

The algorithm does not use derivatives, and converges much more slowly that the *Local Quasi Newton* method, even for problems with only one optimum point.

A further parameter accessible in the advanced settings governing the behaviour of this optimiser is **minboxsize**. This specifies the minimum relative size of the subregions created during the optimisation, and can vary from 0 to 1. Its default value is zero, in which case the optimiser is allowed to split the parameter space indefinitely thus allowing it to (eventually) find the global optimum. For positive values, the minimum box size will be bounded below thus preventing the optimiser to do a systematic search, but forcing it to cover the entire parameter space more quickly, thus potentially making it faster to locate the global optimum.

This parameter therefore allows the user to exploit a-priori knowledge of the smoothness of the *objective function*. Hence **minboxsize** can be set to larger values if the *objective function* is known to vary slowly throughout the parameter space, while it is advised to keep this parameter to zero if there is no a-priori knowledge of the smoothness of the function, e.g. the *objective function* can contain sudden variations or random "spikes" likely to contain interesting optima.

2.5.3 Calculation of the derivatives

By default, derivatives are calculated numerically with a finite difference technique, in conjunction with perturbation techniques, which greatly reduce the number of function evaluations required to calculate the derivatives with respect to all the parameters.

However there is also the option to calculate the derivatives analytically (Fimmprop only). This option is available in the *Advanced Settings* panel. With this option enabled, the derivatives will be calculated using an approximate analytic technique. At the moment, though it can only be used when using one of the *predefined objective functions*. Moreover the approximation may not always be accurate enough. This accuracy is typically of the same order as that of the field calculations, so it is advisable to use it only if the local modal basis sets of each section in the *FIMMPROP Device* are reasonably complete.

As the derivatives are found for every perturbed parameter separately, the computational time of the derivative calculation may remarkably increase if longer length of the device is affected by the perturbation of any of the parameters.

For example, if we optimize the length dependent width of irregular taper (see demo 460), using of the linear interpolation of z-dependent width will result in much faster calculations than the spline interpolation. It happens, because the perturbation of one of the intermediate width parameters will modify waveguide widths in a small part or in whole length of the taper, resulting in shorter or longer derivative calculations.

2.5.4 How to choose the optimisation method

As the *global* optimisation method is much slower than the *local* one, it is only advisable to use it if:

- The number of parameters is relatively small.
- It is likely that the parameter space has many local optima, which would prevent the local optimiser form making any progress
- You have no idea of a good initial guess, since the global optimiser does not use the initial parameter values set in the *Independent Variables* settings.

As a rule of thumb, if you are not sure of the above criteria, it is best to try using the *local* optimiser first. If no progress can be made, then try the global optimiser. Run it until you see a significant function improvement. Then update the initial parameter values to the best point found so far by pressing **Select Best** and then **Update Initial**, and run the *local* optimiser.

Chapter

3

KALLISTOS Interface Reference

Refer to this chapter for a detailed explanation of each feature in KALLISTOS.

3.1 The Main Panel

3.1.1 The toolbar

- View Source Node **source node/view source node** – This sends a command to the application to show the device

- Set Source Node **source node/set source node** – This tells the optimiser the device to be optimised

- Edit Settings **optimiser/Edit settings** – This allows you to set-up the optimisation that you wish to carry out on your device

- Edit Settings as text **optimiser/Edit settings as text** – edit the optimiser settings directly via the *text block*, this is convenient for editing complex setups.

- Run **optimiser/run** – This runs the optimisation for the device with the settings you have provided. If this optimisation has already been run with these settings it continues from where it left off.

- Reset **optimiser/reset calc.** – This resets the cache file for the given settings and so causes a future run to start from the beginning.

- Previous Run **optimiser/previous** – This allows the user to select and view an optimisation which has already been carried out within the optimiser on the same node but with different settings(e.g. using a different optimisation method or a different precision)

3.1.2 The Selected Point box

Select Best This selects the previously calculated point with the best function value

Update Initial This updates the initial values with those of the selected point

3.1.3 The Views

Detach Panel Allows you detach any view from the notebook and show in a separate window

The Tree view

Graphic view specific to the *Global* optimisation method. Each branch of the tree represents a subset of the solution space that is being explored.

The Linepath view

Allows you to view the *objective function* and parameter values for each iteration of the optimisation process. You can display/hide parameter lines using the grid on the right of the parameter graph. On both graphs you can set the following cursor modes by clicking on the top left hand corner:

🔍 - Zoom in/out on graph

✥ - Drag graph

↖ - Select a point with the graph

The Hypercube view

Allows you to view several 2D graphs showing the points for pairs of successive parameters

Choose Graphs	Set which *hypercube view* output graphs to display.

The choose graphs dialog

Select All	Checks all columns
Select None	Unchecks all columns

The Data view

Allows you to view the data for each point calculated in the optimisation process in a spreadsheet form

Choose Cols	Set which data output graphs to display
Export to File	Writes table to output file

The choose columns dialog

Select All	Checks all columns
Select None	Unchecks all columns

3.2 The Settings Panel

In most tabs the following are available

Add	Adds a new row to the table
Delete	Deletes the selected row/rows from the table
Copy	Copies the selected row from the table into the clipboard
Paste	Adds the rows stored in the clipboard to the table

Optimiser Settings tab

This tab contains all the main settings for the optimiser

Precision Determines how close is the final solution to the actual optimum.

Maximum Function Evaluations The maximum functions evaluations allowed. The optimisation will stop if this number is reached.

Advanced Settings Gives access to further parameters affecting the behaviour of the optimisers

Independent Variables Tab

This contains a table with the list of variables that you wish to use in the optimisation process

Objective Tab

New… Create a new user defined type

Edit… Edit an existing user defined type

Delete… Delete an existing user defined type

User defined objective box

Set-up your own objective here, specifying a sequence of *Command-Line instructions* and *output variables*, and an *objective function* in terms of the *output variables*.

Add Adds a new row to the table

Delete Deletes the selected row/rows from the table

Copy Copies the selected row from the table into the clipboard

Paste Adds the rows stored in the clipboard to the table

Test tests the selected command(s).