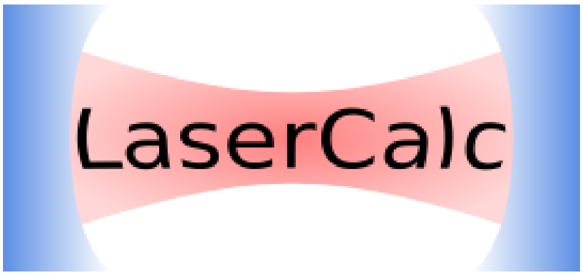


LaserCalc Manual

version 0.1

Udo Eisenbarth (October 2008)

1 Introduction

LaserCalc is a tool to calculate Gaussian laser beams along a path of optical elements. Up to now the program only knows (thin) lenses and distances. However further elements are going to be added soon. Besides the calculation of single-pass setups, the tool also allows to model (ring-)resonators. In this case the fundamental TEM_{00} Eigenmode is automatically calculated. In addition, the stability of the resonator is derived. Finally the mode matcher module tries to find the (nearly-)best solution for the transformation of a beam to given target parameters based on a list of available optical elements. It tries all possible element combinations together with an optimization of the respective distances from each other.

2 Usage

The following section describes all the steps necessary to compile and start the application followed by a description of the different functions and features.

2.1 Compiling

LaserCalc is written in C++ using the C++ bindings of the GTK+ toolkit (GTKmm). So far LaserCalc was only successfully compiled under Linux (Unbuntu 8.04). However it should compile on almost every modern Linux distribution. For compiling the following programs and libraries are necessary:

- GNU C++ compiler (tested with version 4.2.4)
- GTKmm-2.4 (tested with 2.4.16)
- libglademm-2.4 (tested with 2.6.6)

Compiling should be straight forward. Go to the directory ”lasercalc”. The type

```
./configure
```

If any library or program is missing a warning will be printed. If the above step finished without any errors type

```
make
```

This will compile the application. For the last step you have to have administrator privileges. Type

```
make install
```

Without any further switches given on the command line during the configuration the program will be installed in /usr/local/bin. This directory should be in the execution path of the end user. Now you are ready to use the program.

2.2 Starting

The application is started by the user by simply typing

```
lasercalc
```

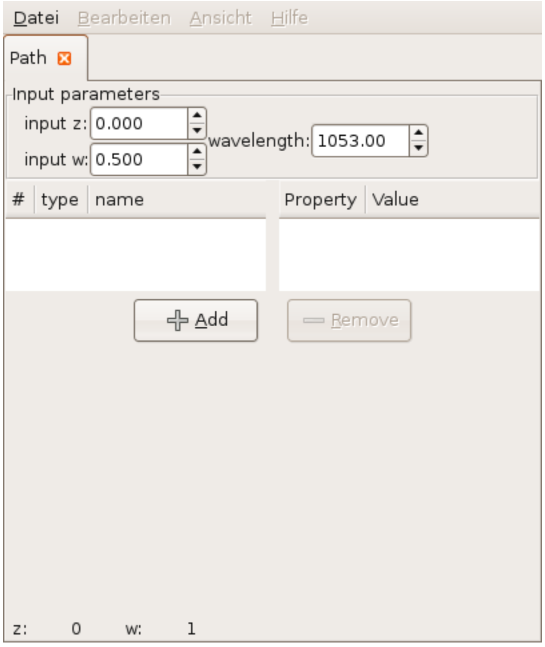
on the UNIX command shell. No additional switches should be necessary. The application starts with an almost empty window. LaserCalc is organized in tabs. Each tab contains a module of the program. So far the modules "Optical Path", "Optical Cavity" and "Mode Matcher" are available. In order to open a new tab go to the "File" menu and select "New". In a submenu one can generate the wanted component. If "Optical Path" is chosen a new tab shows up. This way many calculations can be done in parallel. A click on the close icon (cross) of a tab header closes the respective module.

Figure 1: LaserCalc main window

2.3 Optical Path

The module "Optical Path" calculated the propagation of a Gaussian beam passing a set of optical elements. At first one has to put in the initial beam parameters. A Gaussian beam is defined by its beam waist (w-value) and its distance from this waist (z-value). Hence, if $z = 0$ one would start from the focus position of a laser beam with a focus size of w. Positive z-values mean that one would start with a beam whose focus lies ahead leading to a converging beam. On the other hand, for negative z-values the has already passed the focus resulting in a diverging beam. Further details on Gaussian beams are discussed in the theory section or can be found at `http://en.wikipedia.org/wiki/Gaussian_beam`

As a last parameter the wavelength must be given. It should be noted that all dimensions used in this program are in mm except the wavelength which is set in nm. In order to generate a real path optical elements must be added.



input z: 0.000
input w: 0.500
wavelength: 1053.00
z: 0 w: 1

Figure 2: The module "Optical Path".

For this purpose one has to click on the "Add" button. A dialog appears where the element type to be added can be selected. Currently there are only distances and (thin) lenses. The element will always be appended at the end of the path. In order to check the results it is recommended always to start with a distance. In the following we set up a path consisting of a distance, a lens and another distance. The beam radius along the path is plotted in the lower portion of the window. If the mouse is moved over the graphics

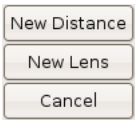
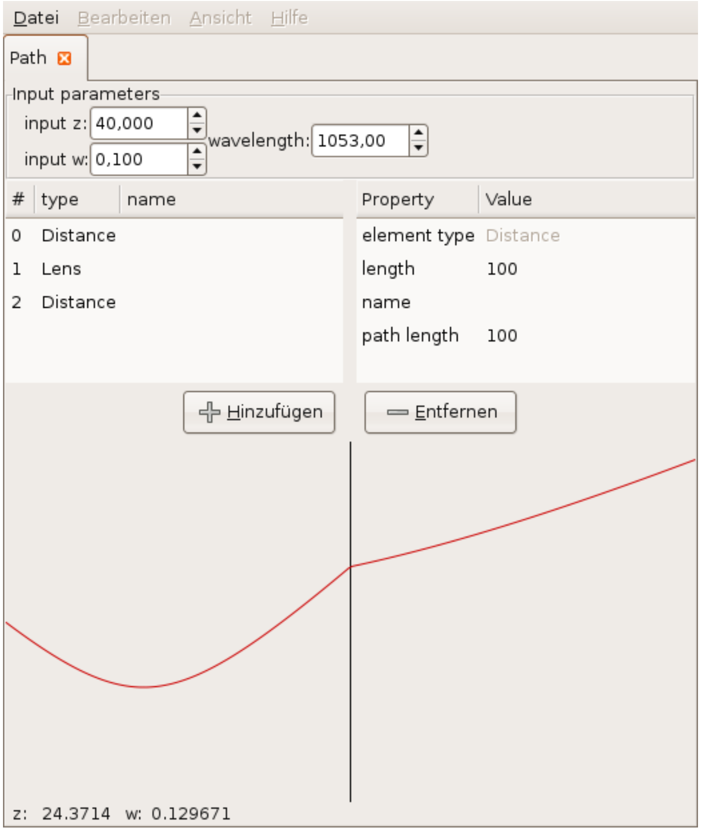


Figure 3: The selection dialog.



input z: 40,000

input w: 0,100

wavelength: 1053,00

#	type	name
0	Distance	
1	Lens	
2	Distance	

Property	Value
element type	Distance
length	100
name	
path length	100

z: 24.3714 w: 0.129671

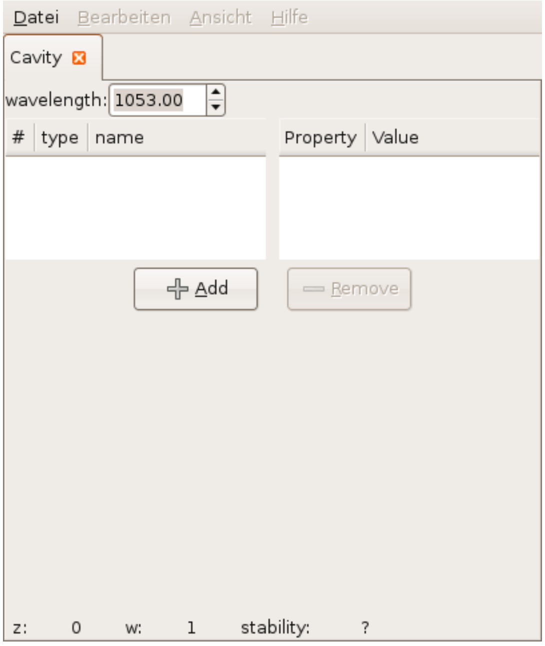
Figure 4: Calculation of an optical path.

window the corresponding values (position and radius) are displayed in the bottom line. The vertical black lines denote the position of optical elements (the lens position in the given example).

If a path is set up it is possible to change the parameters of the optical elements. In the middle of the window two lists are displayed. On left side we have the path element window listing the elements of the path in the order of propagation. The right list displays the properties of a selected element. All non-greyed out properties can be edited by clicking on its value. Each element has a "name" property which also shows up in the element list and the graphics window. In case of a distance element the length can be edited. A lens has a length of zero (it is a thin lens) but the focal length can be set. Positive values denote a convex lens while negative values result in a concave lens. If a property is changed the graphics window will be automatically updated. Finally if you want to remove an element select it in the element list and press the remove button.

2.4 Optical Resonator

The module "Optical Resonator" calculates the TEM_{00} Eigenmode of a ring resonator at a given wavelength. The user interface looks quite similar to the optical path module. However only the wavelength has to be given. Since the Eigenmode is automatically calculated there are no input beam parameters to be defined. The element list works the same as for an optical path. The only difference is that it represents a closed optical path.



wavelength: 1053.00

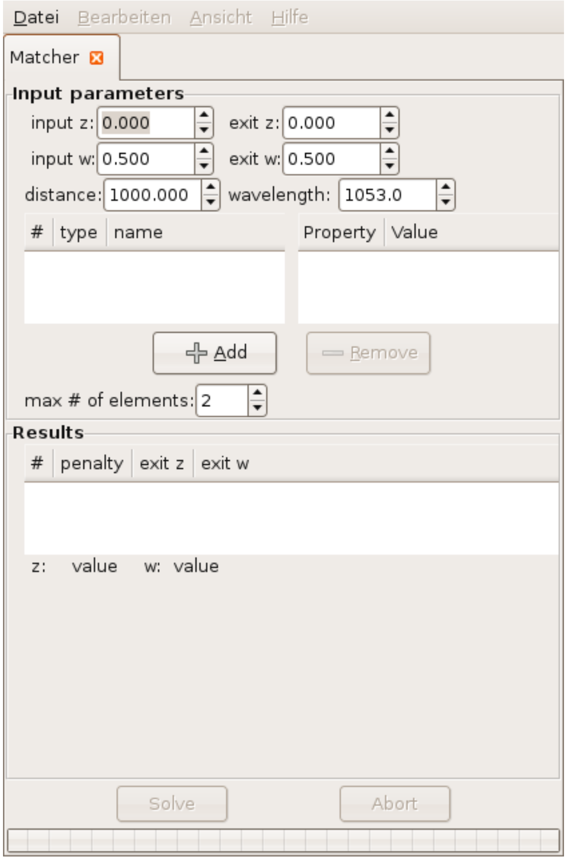
z: 0 w: 1 stability: ?

Figure 5: The module "Optical Cavity".

At the bottom line an additional value "stability" is calculated. This value gives an idea of how stable an optical configuration will be. A stability value of 0% represents an unstable resonator. In this case the plotted curve is meaningless. For real resonators a stability value of more than 60% is desirable. Otherwise the setup would be hard to align and would react strongly to any external disturbances (vibrations, air fluctuations,. . .).

2.5 Mode Matcher

The Mode Matcher module is a very handy tool for transforming one Gaussian beam parameter into another one. In reality the problem often occurs that a given beam (generated by a laser for example) has the wrong size and focus position in order to be fed in a resonator, an optical fiber or any other optical instrument. Another application would be to transport a beam over a certain distance without changing its beam parameter to much. In these cases one usually sets up an optical path consisting of different lenses (in order to realize a magnifying telescope for example). Finding the optimum solution with a given set of lenses can sometimes be quite a time-consuming task. In this case one would use the optical path module and play around with distances and focal lengths until the desired output beam parameter is achieved.



input z: 0.000
exit z: 0.000
input w: 0.500
exit w: 0.500
distance: 1000.000
wavelength: 1053.0

#	type	name

Property	Value

max # of elements: 2

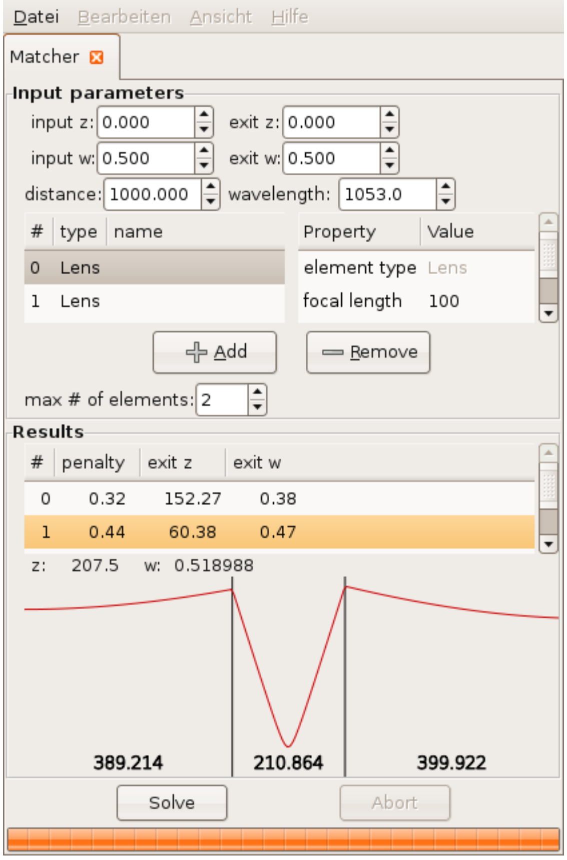
#	penalty	exit z	exit w

z: value w: value

Figure 6: The module ”Mode Matcher”.

In principle there is an analytical solution for this problem which can always be solved with two lenses (sometimes only with one). However there are usually only lenses with standard focal lengths available. The purpose of the mode matcher module is to find the (nearly-) best solution for a given set of optical elements. Usually this non-perfect solution is still sufficiently good enough for the given problem.

In the user interface the input and exit parameters need to be entered. Furthermore the wavelength and the total distance of the optical path. This total distance would normally be the distance from the laser output and the input of the optical instrument. Within this distance LaserCalc tries to put in the optical elements. The middle part of the user interface again contains the element list and the property window. However, in this case the element list does not represent an optical path. It is just a list of elements that can be used by the optimization algorithm. Usually this list should only be filled with lenses. The corresponding distances for the optimum solution are calculated internally. In very rare cases one could also think about adding a distance to this list. This would lead to an optical setup with a fixed length in the path and could possibly be used to provide a minimum separation between optical elements.



Input parameters

input z: 0.000
exit z: 0.000
input w: 0.500
exit w: 0.500
distance: 1000.000
wavelength: 1053.0

#	type	name
0	Lens	
1	Lens	

Property	Value
element type	Lens
focal length	100

max # of elements: 2

Results

#	penalty	exit z	exit w
0	0.32	152.27	0.38
1	0.44	60.38	0.47

z: 207.5 w: 0.518988

Figure 7: The Matcher module with a calculated solution.

The presented case shows a calculation of 1:1 transformation of a laser beam over 1000mm (the standard values). In the element list there were two lenses given with a focal length of 250mm. Pressing the solve button start the calculation which can take quite a while depending on how many elements are on the list and the computer performance of course. LaserCalc tries all combinations of the given elements up to a maximum number of elements in the path which can be entered in the particular field. In the presented case the program delivers four solutions (two solutions with one lens each and two with a combination of both). The solutions are listed with the order of their quality which is given by the penalty value. The penalty value is calculated from the achieved exit parameters in comparison to the targeted exit parameters as entered above. A click a particular solution plots the corresponding beam path with its calculated distances.

Please note that the solutions differ from run to run because of the internal solving process LaserCalc uses a simulated annealing algorithm which is based on random numbers. Therefore the derived solutions are equal only in a certain accuracy level. I doubt, try to run the solver more than once with the same input values to get feeling for the variations.

3 Theory

to be written yet.