signature supervisor

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMA THESIS

Radiation-flux-conservative beam shapes for laser-fluid coupled problems in finite volumes

executed to obtain the academic degree
Diplom-Ingenieur (Dipl.-Ing.) under the supervision of

Univ.Prof. Dipl.-Phys. Dr.-Ing. Andreas Otto

at the
Institute of Production Engineering and Photonic Technologies
Getreidemarkt 9, BA,
9th floor

submitted to the Vienna University of Technology
Faculty of Mechanical and Industrial Engineering

by

Julian Kiesenhofer, BSc

September 23, 2020

signature student

Declaration in Lieu of Oath

I hereby declare to be the sole author of this thesis and that no part of my work has been previously published or submitted for publication. I certify, to the best of my knowledge, that this thesis does not infringe upon anyone's copyright nor violate any other material from the work of others. All knowledge arising from external sources has been fully acknowledged below under standard referencing practice.

Julian Kiesenhofer, BSc

Vienna, September 23, 2020

Abstract

A finite volume code, written in C++, is capable of computing the laser material processing physical phenomena. To explain the whole structure of the light-matter interaction as a transient process, a detailed numerical solution for the propagation of beams must be provided in the first place.

Since we are interested in the molten material's fluid dynamics, which are modeled on a multiphase volume-of-fluid approach, and the laser physics, paraxial laser beam models, are coupled with the fluid problem, it is also advantageous to describe the beam propagation using finite volume methods.

Due to the fact, that there is a large discrepancy between the necessary grid size for the fluid problem and laser physics, some problems concerning energy conservation can arise. The radiation flux-conservation can affect the numerical stability of the absorption model of the laser-fluid coupled field. The theoretical background of this problem is introduced, the models are derived, argued mathematically and the results of these models are presented in this work.
Different models that describe the radiation flux through a finite volume grid are presented and discussed. The models have been implemented in OpenFoam® and the results were visualized in Paraview.

The source code of the multi-physical laser material processing simulation software was developed by IFT Vienna University of Technology and is their intellectual property.

Contents

«Philosophy is written in that great book
whichever lies before our eyes — I mean the universe —
but we cannot understand it if we do not first learn the language
and grasp the symbols, in which it is written.
This book is written in the mathematical language,
and the symbols are triangles, circles, and other geometrical figures,
without whose help it is impossible to comprehend a single word of it,
without which one wanders in vain through a dark labyrinth.»
— Galileo Galilei

Nomenclature

$\boldsymbol{S}_f$	Surface area vector
$\boldsymbol{u}_f$	Face velocity
Δ	Laplace operator
ϵ', ϵ''	Real, Imaginary part of the permittivity
ϵ_0	Vacuum permittivity
$\Im$	Imaginary part of a complex number
λ	Wavelength
$\langle\vec{S}(\vec{r})\rangle$	Time-averaged Poynting vector field
$\mathbb{L}_m^l$	Laguerre polynomials
$\mathbb{M}^2$	Beam quality factor
∇	Nabla operator
Φ	Optical potential
ϕ_f	Volumetric face flux
ϕ_o	Energy face flux
$\Phi_{l,m}$	Phase function of the Laguerre-Gauss modes
$\Re$	Real part of a complex number
θ, ϕ	Elevation, Azimuth angle
$\tilde{A}^{(i)}$	Projected area of cell i

φ	Velocity potential
$\vec{\tilde{x}}$	Absolute coordinates, $[\tilde{x}, \tilde{y}, \tilde{z}]^T$
$\vec{e_l}$	Unit vector along the optical axis
$\vec{n}(\vec{r})$	General laser direction field
$\vec{n}'(\vec{r})$	Analytical model for the energy flux direction
$\vec{n}_\Phi(\vec{r})$	Optical potential model for the energy flux direction
$\vec{x_0}$	Position vector to the beam origin
$\vec{r}, \vec{x}$	Local coordinates, $[x, y, z]^T$
$\zeta(z)$	Gouy phase
A^*	Complex conjugate of A
$I_f(\vec{r})$	Face beam intensity
$I_g(\vec{r})$	Intensity field of the Gaussian beam
P	Optical power of a laser
$R(z)$	Wave front curvature along the optical axis
U_m^l	Wave function of a Laguerre-Gaussian beam
$V^{(i)}$	Finite volume of cell i
$W(z)$	Beam caustic
W_0	Beam waist
z_0	Rayleigh length

1. Introduction

Laser material processing has gained a significant level of maturity and industry acceptance. This is used for cutting, drilling, welding, hardening, and different types of metal surface preparation across a vast range of industrial industries, including the automobile and aerospace sectors, the shipbuilding industry, the microelectronics industry, and the medical device industry, to name just a few.

The importance of numerical simulations in the processing of laser materials is continuously increasing for various applications in science and industry. These simulations offer a deeper interpretation of the process for analytical purposes, e.g. by modifying various material or process parameters independently and evaluating their effect on the results. Advantages involve the prospect of getting an inside view or findings over very small time scales which are difficult or impossible to experimentally grasp. Simulations serve as a compatible tool for industrial interests in developing new production steps, whole process chains, or even products [12].

In many industrial applications, simulations are state-of-the-art as opposed to laser material processing, e.g. aerodynamics in the automotive or aerospace industry. This is due to the complexity of processes coupled with laser-matter, and the fact that involves many different physical phenomena [14]. To reduce complexity, state-of-the-art models decouple much of the system or even neglect some when examining a specific process. In this way, standard models can only analyze certain processing effects. For multi-physical methods, each physical subject must be applied in such a way that no contradictions occur with other physical rules, e.g. numerical calculation of a laser-intensity field that is not precisely conservative in energy due to energy flux assumptions. This work gives an overview of the calculation methods and presents a new method for calculating the energy flux of three-dimensional beam shapes in finite volumes, with respect to the capability to be coupled in multi-physical simulations like laser material processing.

2. Theoretical Background: Optics and Fluid Dynamics

In the following chapter, the theoretical background which is needed in this thesis is given. As well the photonics background, which is typically beam optics in laser material processing, and the fluid dynamics background is explained. In our case especially the potential flow theory is introduced. The model of the multiphysics simulation is a multi-phase flow problem, the theoretical background of this theory is briefly discussed. The energy transport of the laser-fluid coupling and its model is discussed in detail in chapter 3.

2.1. Beam optics

The nature of light and lasers, Photonics, is subdivided into related theories that describe laser mechanisms on a different level of complexity. Since we are interested in laser material processing, the right level of complexity, at least the highest possible level in the sense for macroscopic simulations, is beam optics. The most fundamental ideas and laws of beam optics are introduced in the following chapters.

2.1.1. The paraxial Helmholtz equation

In the beginning, the meaning of paraxial approximation, or small-angle-approximation, needs to be explained. In an optical system, the propagation of light is often described by Gaussian optics and ray tracing. A so-called paraxial ray is a ray which encloses a small angle with the optical axis in an optical system [2]. The generally first-order ray tracing is used in Gaussian optics, where the three important approximations are used,

$$\begin{aligned} sin\theta &\approx \theta, \\ tan\theta &\approx \theta, \\ cos\theta &\approx 1. \end{aligned} \tag{2.1}$$

In some cases, paraxial approximation means the second-order Taylor series as well, which is given by

$$cos\theta \approx 1 - \frac{\theta^2}{2}. \tag{2.2}$$

Consequently, a wave can be called paraxial, if its wavefronts are perpendicular to paraxial waves. The idea of that approximative approach is to start with a carrier plane wave $A\exp(-jkz)$ and to modulate its complex envelopment in space. $A(\vec{r})$ is a slowly varying function of space, and the complex amplitude can be described by,

$$U(\vec{r}) = A(\vec{r})\exp(-jkz). \tag{2.3}$$

$A(\vec{r})$, respectively its derivative by z, has to vary slowly with respect to the wavelength $\lambda = 2\pi/k$ of the beam. The general complex wave equation can be stated,

$$\frac{1}{c}\frac{\partial^2 U(\vec{r})}{\partial t^2} - \nabla^2 U(\vec{r}) = 0. \tag{2.4}$$

After introducing the wavenumber $k = 2\pi f/c = \omega/c$, we can rewrite the wave equation,

$$\nabla^2 U(\vec{r}) + k^2 U(\vec{r}) = 0, \tag{2.5}$$

which is know as the general Helmholtz equation. The paraxial beam uses, as already named, the assumption, that $A(\vec{r})$ varies slowly with z with respect to the wavelength of the beam. This relation reads

$$\frac{\partial A(\vec{r})}{\partial z} \ll kA(\vec{r}). \tag{2.6}$$

If the first derivative by z varies slow, the derivative of the first derivative must vary slow as well,

$$\frac{\partial^2 A(\vec{r})}{\partial z^2} \ll k^2 A(\vec{r}). \tag{2.7}$$

Inserting this relation into the general Helmholtz equation leads to the paraxial Helmholtz equation and which by expressed by,

$$\nabla_T^2 A(\vec{r}) - j2k\frac{\partial A(\vec{r})}{\partial z} = 0, \tag{2.8}$$

where $\nabla_T^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$ is known as the transverse Laplacian operator [2]. In the following two chapters solutions of this equation are introduced.

2.1.2. Fundamental Gauss mode

The fundamental mode or a Gauss mode has a central meaning in laser material processing [4]. Especially the straight-forward integration methods in the later finite volume integration has advantages and gives a good accuracy in its energy conservation. This is given by the fact, that the Gaussian function can be integrated by the Error Gauss function, even in two dimensions. In the following chapters, the theoretical background for the beam shape description is introduced. The theories are mathematically defined in polar coordinates in contrast to the finite volumes afterward. The problem of its energy flux direction will be discussed as well in a later step.
In the following equations, the properties of a Gaussian beam will be explained [2].
The complex amplitude of the beam reads

$$U(r,z) = A_0 \frac{W_0}{W(z)} \exp\left[-\frac{2r^2}{W^2(z)} - jkz - jk\frac{2r^2}{2R(z)}\right] \exp\left[-j\tan^{-1}\frac{z}{z_0}\right], \tag{2.9}$$

and its real part can be seen in figure 2.1. The optical intensity, as a function of space, is given by

$$I_g(r,z) = \left|U(r,z)\right|^2 = \frac{2P}{\pi W^2(z)} \cdot \exp\left[-\frac{2r^2}{W^2(z)}\right]. \tag{2.10}$$

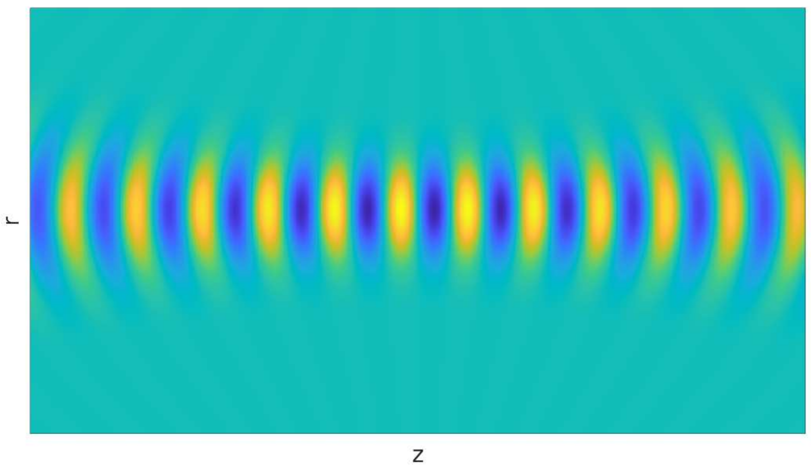


Figure 2.1.: Wave function $U(r,z)$ of a Gaussian beam (generated in MATLAB R2019b, code (wavefront.m) appended in chapter A.)

The beamwidth as a function of z is the radial position of the beam where the intensity dropped by a factor of e^{-2} compared to the optical axis.

$$W(z) = W_0 \cdot \sqrt{1 + \left(\frac{z}{z_0}\right)^2} \tag{2.11}$$

The relation of the Rayleigh range z_0 and the beam waist W_0 is known as

$$W_0 = \sqrt{\frac{\lambda z_0}{\pi}}. \tag{2.12}$$

The beam radius is the curvature radius of the wavefront and given by the following equation [2].

$$R(z) = z \cdot \left[1 + \left(\frac{z}{z_0}\right)^2\right] \tag{2.13}$$

Figure 2.2 illustrates the phase function of a Gaussian beam in two dimensions. The third dimension gives the value of the phase which is the z-axis of the graph. Additionally, the magnitude of the phase function is plotted in this figure for illustrative reasons. The real phase function is point-symmetric.
Especially in the multi-physics of laser material processing the optical power of the beam is of interest due to the necessity of energy conservation [6]. The property

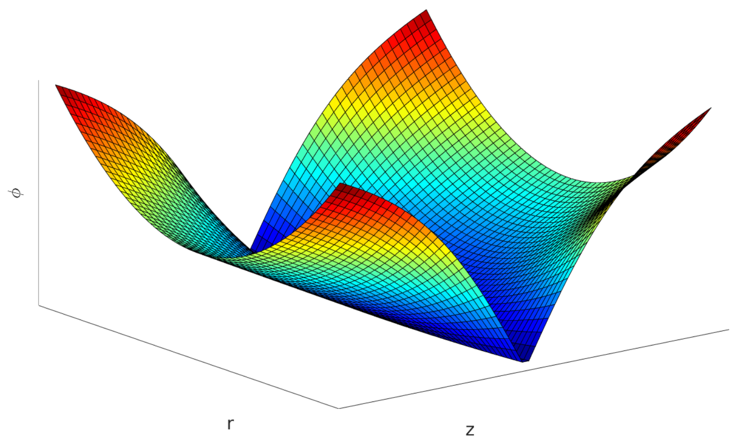


Figure 2.2.: Magnitude of Gaussian electric field phase or wave phase function (generated in MATLAB R2019b, code (opticalPotential.m) appended in chapter A.)

of the optical power is known as the infinite surface integral over a perpendicular plane to the optical axis

$$P = \int_0^\infty I_g(r,z) 2\pi r \mathrm{d}r. \tag{2.14}$$

After inserting equation (2.1) and performing an analytical integration we yield at any position of z:

$$P = \frac{1}{2} I_0 \pi W_0^2. \tag{2.15}$$

2.1.3. Laguerre-Gauss modes

The solution for Laguerre-Gaussian beams [2], which is another solution of the paraxial Helmholtz-equation, is given by,

$$\begin{aligned} U_{l,m}(r,\theta,z) = A_{l,m}\left[\frac{W_0}{W(z)}\right]\left(\frac{r}{W(z)}\right)^l \mathbb{L}_m^l\left(\frac{2r^2}{W^2(z)}\right) \\ \cdot \exp\left(-\frac{r^2}{W^2(z)}\right) \times \exp\Big(\Phi_{l,m}(r,\theta,z)\Big). \end{aligned} \tag{2.16}$$

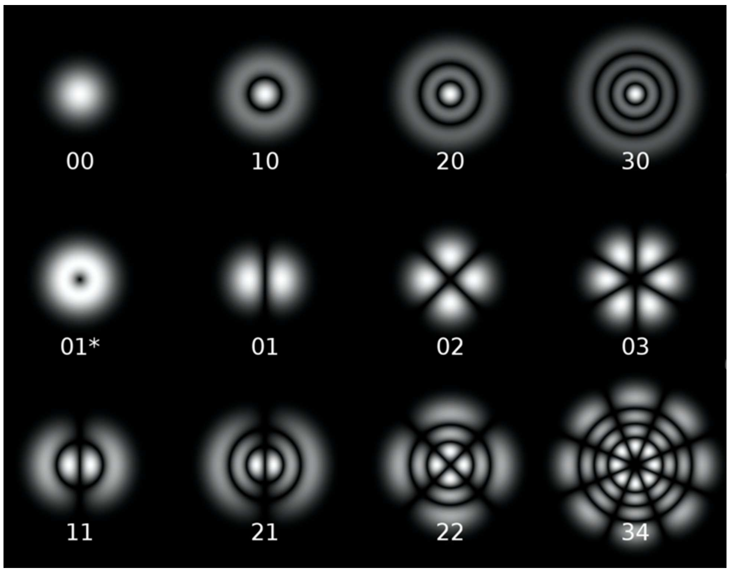


Figure 2.3.: Laguerre-Gauss beam modes (l,m) by squaring equation (2.16)

Whereby,

$$\begin{aligned}
\Phi_{l,m}(\vec{r}) &= -j(kz+l\theta) - jk\frac{r^2}{2R(z)} + j(l+2m+1)\zeta(z),\\
\vec{r} &= (r,\theta,z)^T,\\
W(z) &= W_0\sqrt{1+(\frac{z}{z_0})^2},\\
R(z) &= z\Big[1+(\frac{z_0}{z})^2\Big],\\
\zeta(z) &= \arctan\frac{z}{z_0},\\
z_0 &= \frac{\pi W_0^2}{\lambda \mathbb{M}^2}.
\end{aligned} \tag{2.17}$$

The given solution exists in cylindrical coordinates, which is obtained by using the separation-of-variables technique. The transverse modes can be seen in figure 2.3. The Laguerre Polynomials are given by,

$$\mathbb{L}_m^l(x) = x^{l+m}e^{-x}\frac{x^{-l}e^x}{m!}\frac{d^m}{dx^m}. \tag{2.18}$$

Modern laser source manufacturers can build a wide variety of different beam shapes with nearly ideal shape. Most simulations are a very good reference if the ideal shapes like Gaussian (circular or elliptical) or the Tophat beam is calculated since many other shapes can be created by a superposition of them. In figure 2.4 intensity profiles of two Gaussian, a Tophat, and a multi-mode beam can be seen, where black color means zero and white color maximum intensity.

Since the Helmholtz equation is a linear partial differential equation, superpositions of its solution are solutions themselfs. Consequently, a wide variety of different beam shapes can be constructed numerically if the higher-order fundamental modes are used [9].

2.2. Lambert-Beer's law

If the permittivity is a complex function, it can be described by,

$$\epsilon_r'(\omega) + i\epsilon_r''(\omega) = 1 + c\frac{N_A(\alpha'(\omega)+i\alpha''(\omega))}{\epsilon_0}. \tag{2.19}$$

Using the Maxwell's equation for an isotropic scalar media leads to,

$$n(\omega) + ikn(\omega) = \sqrt{1 + c\frac{N_A(\alpha'(\omega) + i\alpha''(\omega))}{\epsilon_0}}. \tag{2.20}$$

The first order Taylor series approximation reads $\sqrt{1+x} \approx 1+x/2$. Consequently, we can approximate equation (2.20) by,

$$n(\omega) + ik(\omega) = 1 + c\frac{N_A(\alpha'(\omega) + i\alpha''(\omega))}{2\epsilon_0}. \tag{2.21}$$

Separating real and imaginary part gives the following function for the absorption index function,

$$k(\omega) = c\frac{N_A\alpha''(\omega)}{2\epsilon_0}. \tag{2.22}$$

If we consider the absorption function $A(\omega)$, whereby d is the thickness of the cell,

$$A(\omega) = 4\pi \log_{10}(e) d\omega k(\omega), \tag{2.23}$$

we can derive Lambert-Beer's law [7].

$$A(\omega) = \frac{2\pi \log_{10}(e) N_A \omega \alpha''(\omega)}{\epsilon_0} cd \tag{2.24}$$

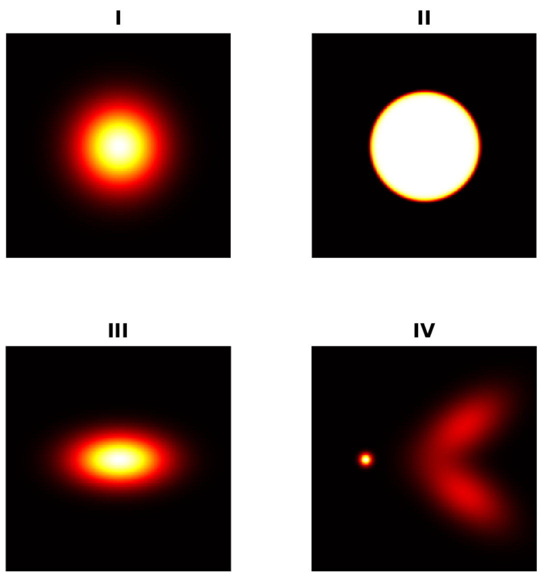


Figure 2.4.: $I(r, z)$ for I: a Gaussian beam; II: a tophat beam; III: an elliptical Gaussian beam; IV: a multi-mode beam (generated in MATLAB R2019b, code (BeamShapes.m) appended in chapter A.)

This law has the capability to connect the physics of optical fields with the idea of a multiphase flow, if the molar concentration c of equations (2.19-2.24) and the phase fraction α_k of equation (2.27) is considered as the same quantity.

2.3. Multiphase flow: Volume of fluid approach (VOF)

In laser material processing the material is generally considered as a fluid. Due to the fact, that during the processes phase changes arise and a surrounding material is part of the process, a multiphase flow model is used for computing the problem. The word 'phase' can be understood as solid, liquid, vapour or plasma state of the material. The prefix 'multi' means several phases arise in one flow. The classification of such a flow is made if the present phases are either dispersed or separated. If particles or droplets present one phase and it has many individual interfaces, it is called a dispersed flow. A separated flow, on the other hand, has only a few interfaces. The most important quantity to characterize a multiphase flow is the dispersed-phase volume fraction [1]:

$$\alpha_d = \frac{\sum_{i=1}^{N_d} V^i}{V} \tag{2.25}$$

If a model for a multiphase problem is chosen, it is useful to determine a dimensionless number, giving the ratio of a dispersed-phase timescale and a continuous-phase timescale. The turbulent Stokes number is defined by,

$$St_T = \frac{\tau_d}{\tau_T}, \tag{2.26}$$

wherein the index d means dispersed phase and index T means the turbulent timescale. In the sense of particle trajectory, the Stokes number explains, if the particle follows the continuous phase flow. The limits of this number can be seen in figure 2.5.

Many models for the computation of a multiphase flow are available. The most important ones can be listed as the Euler-Lagrange model, the Euler-Euler model, the mixture or algebraic slip model, the volume-of-fluid (VOF) model, and the porous-bed model.

Type	Definition	Implementation
One-way	Dispersed phase senses continuous phase, but the continuous phase is unaffected	Particle tracking may be done in a post-processor
Two-way	Dispersed phase senses continuous phase and continuous phase senses dispersed phase	The presence of the dispersed phase should be reflected in the governing equations of the fluid
Four-way	Dispersed particles interact	A model for particle-particle interactions should be included

Table 2.1.: Various types of coupling between the dispersed phase and the continuous phase

The volume-of-fluid approach has proven to be a good model to compute the fluid dynamics of a molten weld pool [12].
The volume-of-fluid model, or VOF-model, can be used for large bubbles in liquids, stratified flow, and, like in laser material processing, free-surface flows. The Euler-Euler model, which tracks the interface of the different phases, can describe the flow of two or more phases. Numerically, it is the model for the VOF-approach, whereby VOF requires well-resolved interfaces [1].
Euler-Euler treats a multiphase flow as fully inter-penetrating quasi-fluids. As an assumption, the flow quantities are averaged by its volume fraction of α_k, which is illustrated in figure 2.6. This can be understood as a small volume, which is way smaller than the large-scale flow structure, but larger than volumes of the smallest dispersed phase fractions. This fraction does not say anything about the flow itself but is necessary as a closure model. In computational fluid dynamics problems, it can be calculated by the distribution of phases and the size of the computational

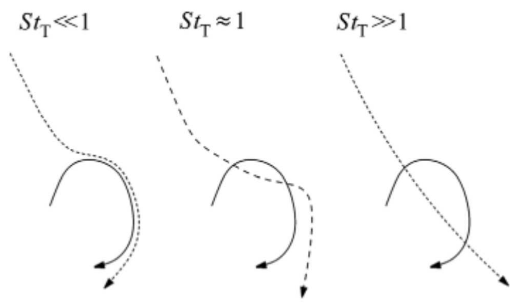


Figure 2.5.: Stokes numbers limits, dispersed phase trajectory (dashed line) and turbulent eddy (solid line)

volume. The governing equations, which describe the VOF-approach are given by [1],

$$\sum_k \alpha_k = 1, \tag{2.27}$$

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \frac{\partial \alpha_k \rho_k U_{i,k}}{\partial x_i} = -\sum_{l=1}^{N} (\dot{m}_{kl} - \dot{m}_{lk}), \tag{2.28}$$

$$\frac{\partial \alpha_k \rho_k U_{i,k}}{\partial t} + \frac{\partial \alpha_k \rho_k U_{i,k} U_{j,k}}{\partial x_j} = -\alpha_k \frac{\partial p}{\partial x_i} + \frac{\partial \alpha_k \tau_{ij,k}}{\partial x_k} + \alpha_k \rho_k g_i + F_{i,k}. \tag{2.29}$$

2.4. Potential flow theory

A potential function φ can be defined as a continuous function which fulfills the basic laws of fluid dynamics: mass and momentum conservation. An incompressible, irrotational and inviscid flow has to be assumed. As a vector identity, we can state:

$$\nabla \times \nabla \varphi = 0. \tag{2.30}$$

An irrotational field is defined by,

$$\nabla \times \vec{v} = 0. \tag{2.31}$$

Consequently, $\vec{v}$ can be described by a gradient of a scalar potential,

$$\vec{v} = \nabla \varphi. \tag{2.32}$$

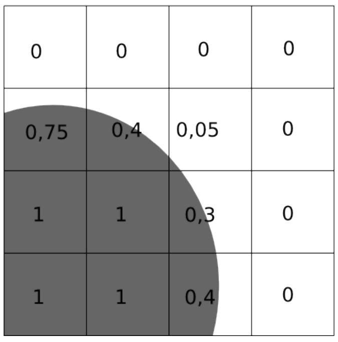


Figure 2.6.: Two-dimensional scheme of the VOF-approach in a 4x4 grid

The continuity equation requires that $\vec{v}$ is solenoidal. Inserting (2.32) into the condition of a solenoidal vector field leads to the Laplace's equation.

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0 \tag{2.33}$$

Since the present numerical model for laser material processing is discretized by finite volumes, we are interested in a numerical solution of the Laplace's equation with the method of finite volumes as well.
For this purpose, the Poisson's equation has to be solved.

$$\nabla^2 \varphi = \nabla \phi_f \tag{2.34}$$

The volumetric face flux is denoted as ϕ_f and is computed by,

$$\phi_f = \boldsymbol{u}_f \cdot \boldsymbol{S}_f. \tag{2.35}$$

By interpolating the velocity field $\vec{v}$ of the surrounding cell values, the face velocity vector $\boldsymbol{u}_f$ is obtained for one face. Given by the normal vector of the face, scaled with its area, the surface area vector $\boldsymbol{S}_f$ can be denoted [10].

3. Laser-fluid coupled transport problems

3.1. Coordinate problem

A domain for a multi-physical problem is given in Cartesian coordinates. The position vector of the domain is called $\tilde{\vec{x}}$. The given idea is sketched in figure 3.1. To find an algebraic solution of the problem the domain is discretized into finite volumes that are aligned to the coordinates of the Cartesian coordinate system. In this work only finite volumes with the geometry of orthogonal hexahedrons are taken into account. The general physical problem is described by the continuum's mechanics for a multiphase flow. Consequently, a description for the beam shape in finite volumes has to be found as well to be coupled with the general problem. In some cases, the optical axis is not perpendicular to any finite volume's surface and either way, the mathematical description of the intensity field is defined in polar coordinates. Coordinate transformations are necessary for obtaining the solution of the intensity field.

3.2. Absorption model for laser-fluid coupled problems

The general transport equation can be described as follows [13],

$$\frac{\partial}{\partial t}(\rho\psi) + \nabla \cdot \phi\psi - \nabla \cdot \Gamma\nabla\psi = S_{\psi}. \tag{3.1}$$

The VOF-approach gives the values of the single phases of the continuum by a convective transport equation, whereby α_i corresponds to the i-th volumetric part of the continuum and u gives the vector field of the velocity [3].

$$\frac{\partial \alpha_i}{\partial t} + u \cdot \nabla\alpha_i = 0 \tag{3.2}$$

Thus, it can be concluded for every field point that,

$$\sum_{i=1}^{N} \alpha_i = 1, \tag{3.3}$$

where N is the number of the different continuous phases of the flow problem. The question arises if a mixture, containing vapor, liquid, and solid phases, can be considered as a fluid or has solid characteristics. In our model, this problem is solved by defining a threshold, typically 0.5, which is compared to the solid fraction of the continuum. Below this value, the whole mixture is modeled as a fluid, above as a solid body. In the latter sense, a source term is added in the momentum equation to suppress the solid movement. Anyway, multiphase flow, in general, can be subdivided into a solid, liquid, and vapor phase [5].
For laser material processing the intensity field of the laser-fluid problem has to be modeled two-way coupled. Reflections of the laser beam can not be neglected

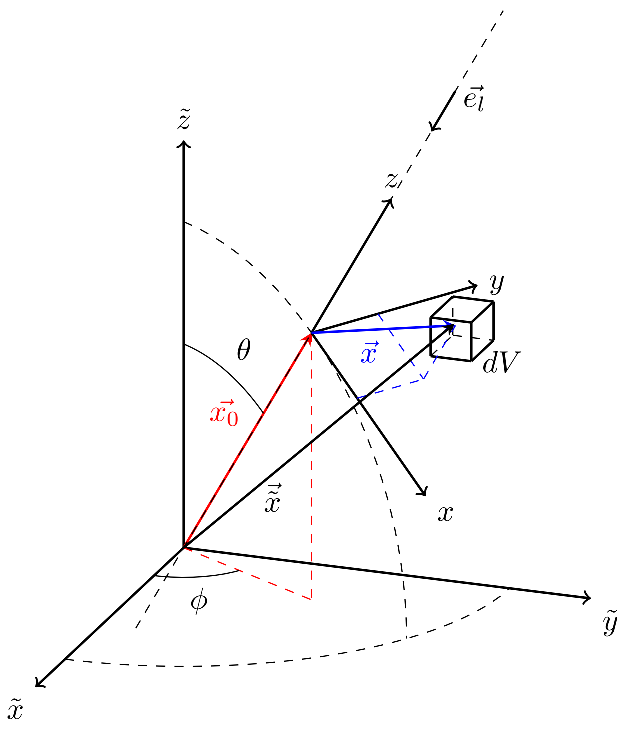


Figure 3.1.: Coordinate problem of the beam description (generated with the LaTex TikZ package, `http://www.texample.net/tikz/examples/the-3dplot-package/`).

and a steadily accumulated solution for the intensity distribution has to be found. This is done by two coupled radiation transport equations.

$$\nabla(n\hat{I}) + (\beta_{BLV} + \beta_{VA} + \beta_{PA} + \beta_{CA}) \cdot \hat{I} = (1 - R)\beta_{BEV} \cdot \hat{I}_T \tag{3.4}$$

$$\nabla(n\hat{I}_T) + (\beta_{BEV} + \beta_{ABS} + \beta_{CA}) \cdot \hat{I}_T = (1 - R)\beta_{BLV} \cdot \hat{I}. \tag{3.5}$$

n	energy flux direction
$\hat{I}$	normalized intensity
$\hat{I}_T$	normalized transmission
R	reflectivity
β_{BLV}	beam leaving vapour
β_{VA}	vapour absorption
β_{PA}	plasma absorption
β_{CA}	cloud absorption
β_{BEV}	beam entering vapour
β_{ABS}	absorption

Table 3.1.: variables transport equation

To obtain an algebraic solution for the set of partial differential equations an integration over the volume has to be performed.

$$\int_V \nabla(n\hat{I})dV + \int_V (\beta_{BLV} + \beta_{VA} + \beta_{PA} + \beta_{CA}) \cdot \hat{I}dV = \int_V (1 - R)\beta_{BEV} \cdot \hat{I}_T dV \tag{3.6}$$

$$\int_V \nabla(n\hat{I}_T)dV + \int_V (\beta_{BEV} + \beta_{ABS} + \beta_{CA}) \cdot \hat{I}_T dV = \int_V (1 - R)\beta_{BLV} \cdot \hat{I}dV. \tag{3.7}$$

The laser material processing PDE's setup is discretized and solved numerically by the method of finite volumes in OpenFoam [11].

3.3. Idea of rayf in flux modeling

Applying the Gauss's theorem, the volume integration of the divergence of the energy flux vector field can be rewritten as,

$$\int_V \nabla(n\hat{I})dV = \int_A \hat{n}(n\hat{I})dA, \tag{3.8}$$

where $\hat{n}$ is an unit vector perpendicular to the domain of V and n is the direction of the energy flux. By knowing that the volume is a hexahedral cell, the surface integral can be calculated by,

$$\int_A \hat{n}(n\hat{I})dA = \sum_{i=1}^{6} \underbrace{\hat{n}_i n A_i}_{rayf_i} \hat{I}. \tag{3.9}$$

The shown rayf, a scalar product of the energy flux vector and the normal vector onto a face multiplied by its area, is calculated for all faces of every cell and is essential for the accuracy of the differential equations' solution.

4. Beam propagation model approaches

In the following chapter, the theoretical background and ideas of the different beam model approaches are given. Since there is a large difference between the necessary grid size of the fluid dynamics and the laser physics in laser material processing certain beam propagation models or beam models are used.
A model for the energy flux direction $\vec{n}$ in field description (localrayf) has to be found.

The general radiative transport equation is derived from the Boltzmann transport formalism and reads,

$$\frac{1}{c}\frac{\partial I_\nu}{\partial t} + \nabla \cdot \vec{n} I_\nu = j_\nu - (\kappa_\nu + \sigma_\nu) I_\nu. \tag{4.1}$$

Laser light, which is usually monochromatic, has only one certain wavelength. Its intensity has no spectrum and we can state [8], $I_\nu = I$.

If we consider a stationary radiation field, we can conclude that the time derivative of the intensity must be zero, $\frac{1}{c}\frac{\partial I_\nu}{\partial t} = 0$.

The model is computed under the assumption, that the medium for propagation is a vacuum and consequently no scattering or absorption occurs, $\kappa_\nu = \sigma_\nu = 0$.

The radiative transport equation can be simplified to,

$$\nabla \cdot \vec{n}(r, z) I(r, z) = 0. \tag{4.2}$$

We know from electrodynamics,

$$\nabla \langle \vec{S} \rangle = \frac{1}{2} \omega_0 \varepsilon_0 \varepsilon''(\omega_0) \vec{E}(\vec{r}) \vec{E}^*(\vec{r}). \tag{4.3}$$

Applied to vacuum ($\varepsilon'' = 0$), we can state,

$$\nabla\langle\vec{S}\rangle = 0. \tag{4.4}$$

The intensity field is defined as follows,

$$I(r,z) = \left|\langle\vec{S}(r,z,t)\rangle\right|. \tag{4.5}$$

An intensity field can be given by a Gaussian beam, if we are interested in paraxial laser beams [2],

$$I(r,z) = \frac{2P}{\pi W^2(z)} \cdot \exp\left[-\frac{2r^2}{W^2(z)}\right]. \tag{4.6}$$

The remaining unknown is $\vec{n}(r,z)$. It can be assumed that this is the direction vector of the time-averaged Poynting vector field $\langle\vec{S}\rangle$. However, this is not so easy to calculate, since the electric field is required for that, $\vec{S} = \vec{E} \times \vec{H}$, $\langle\vec{S}\rangle = \frac{1}{T}\int_0^T \vec{S}\,dt$.

The Gaussian beam is visualized in figure 4.1, where the horizontal axis is the z-axis and the vertical axis is the r-axis.
Equation 4.4 and 4.5 can be used to derive,

$$\langle\vec{S}\rangle = \vec{n}(r,z)I(r,z). \tag{4.7}$$

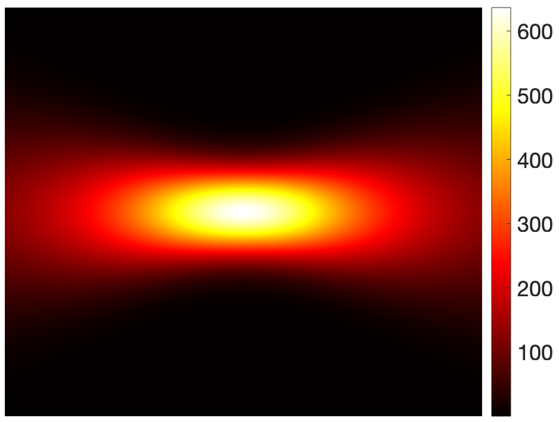


Figure 4.1.: Intensity $I(r,z)$ of a Gaussian beam by equation (4.6) (generated in MATLAB R2019b, code (BeamShapes.m) appended in chapter A).

For the sake of simplicity, the following Matlab experiments are performed in two dimensions. The unit vector $\vec{e}_z$ is assumed for $\vec{n}$.

$$\nabla \cdot \vec{e}_z I(r,z) = 0. \tag{4.8}$$

$$\nabla(\vec{e}_z I(r,z)) = \frac{\partial I(r,z)}{\partial z}. \tag{4.9}$$

Errors concerning the energy conservation arises here, except along the optical axis, because the assumption of the local direction of propagation is correct there.

Consequently, other unit vectors are assumed for the local direction of propagation.

If we are interested in the normal vector on the wavefront of the Gaussian beam, we have to consider the wavefront curvature of the beam.

We know the radius of curvature of the wavefront along the beam axis [2] for $z \in \mathbb{R} \setminus \{-z_R < z < z_R\}$, since the wavefront can be considered as a spherical wave in that region.

$$R(z) = z\big[1 + \big(\frac{z_R}{z}\big)^2\big]. \tag{4.10}$$

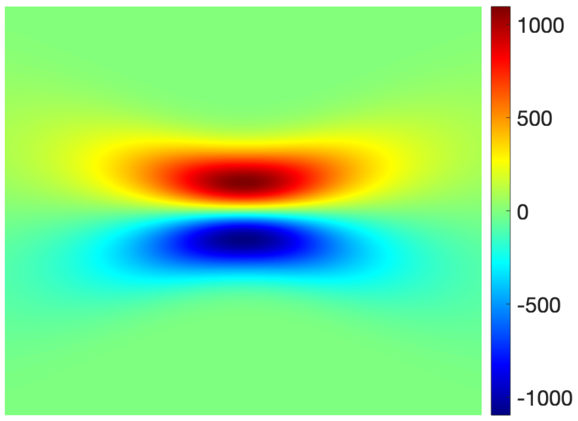


Figure 4.2.: Divergence of the unidirectional flux of the intensity $\nabla(\vec{e}_z I(r,z))$ of a Gaussian beam by equation (4.9) (generated in MATLAB R2019b, code (BeamShapes.m) appended in chapter A).

Consequently, we know the radius of curvature in each field point, which is shown in figure 4.3,

$$R(r,z) = \sqrt{z^2\big[1 + \big(\frac{z_R}{z}\big)^2\big]^2 + r^2}. \tag{4.11}$$

It can be shown, that the vector field $\vec{n}I(r,z)$ of the Gaussian beam is source- and sink-free in a very good approximation and namely solenoidal. Apparently, that field is in its validity range of $R(r,z)$: $z \in \mathbb{R} \setminus \{-z_R < z < z_R\}$ and $r \in \mathbb{R}^+$ a very good approximation of the time-averaged Poynting vector field, whereby $\vec{n}$, by equation (4.12), gives the direction of the energy flux (localrayf). This model for the energy flux direction is used in the following tests to analyze if paraxial beams are source- and sink-free using the model for different grid refinements.

$$\vec{n}(r,z) = \frac{1}{R(r,z)}\big[\vec{e}_r\sqrt{R(r,z)^2 - r^2} + \vec{e}_z z\big]. \tag{4.12}$$

Using equation (4.4) we can compute,

$$\nabla(\vec{n}I(r,z)) = \frac{\partial(n_r(r,z)I(r,z))}{\partial r} + \frac{\partial(n_z(r,z)I(r,z))}{\partial z}. \tag{4.13}$$

Let us apply this analysis to all beam models and consider the numerical radiation flux divergence at different resolutions.

Figure 4.4 - 4.6 show several divergence terms of the flux direction model multiplied by their paraxial beam intensity profiles. The plots show, that the sink or

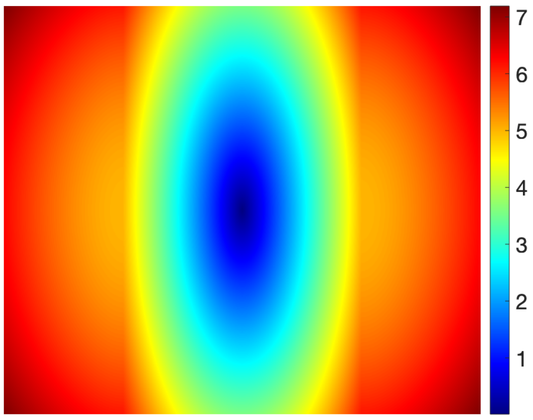


Figure 4.3.: $R(r,z)$ by equation (4.11) (generated in MATLAB R2019b, code (opticalPotential.m) appended in chapter A).

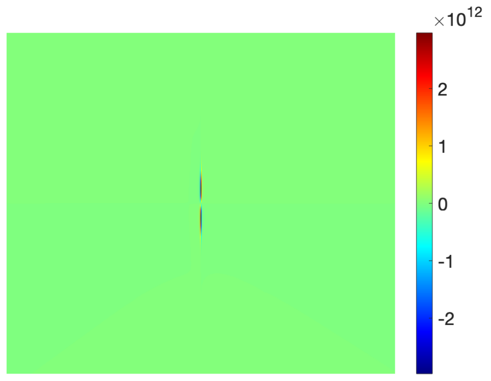


Figure 4.4.: Divergence of the modeled intensity flux $\nabla(\vec{n}I(r,z))$ of a Gaussian beam by equation (4.13) (generated in MATLAB R2019b, code (TEM-modes.m) appended in chapter A).

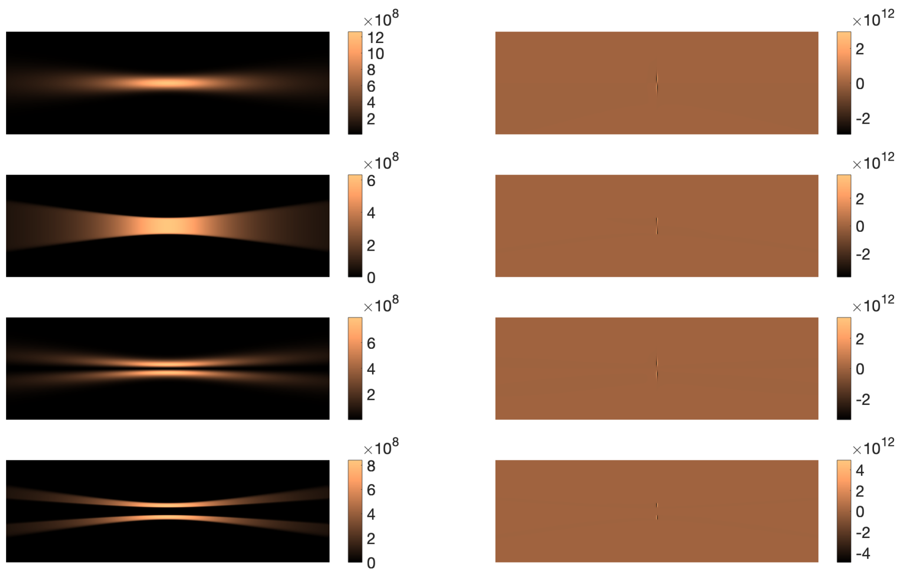


Figure 4.5.: Left: intensity $I(r,z)$ different beam profiles, right: divergence of the modeled intensity flux $\nabla(\vec{n}I(r,z))$ by equation (4.13), Mesh: 1024x1024 (generated in MATLAB R2019b, code (opticalPotential.m) appended in chapter A).

source freeness is strongly dependent on the energy flux of the beam model. If a PDE for the intensity transport is set up, which is meant to be conservative, this has to be taken into account.
The coarser the mesh, the more the field deviates from the freedom from divergence. Two minima and maxima are visible around the origin of the beam, which is reminiscent of quadrupoles. In a homogeneous medium, viewed globally, these artifacts should even out themselves through their point-symmetrical shape, only the local radiation flux equilibrium is not given.
The divergence for all modeled shapes is calculated equally with dependence on z, since it is assumed that the profiles are modeled in a large domain within the far-field solution is sufficient.
Nevertheless, numerical errors vary with z in a general equidistant grid for all shapes in finite volumes. It can be seen, that the energy flux model is in good agreement with the condition of energy conservation in the far-field for any beam shape, but especially near the origin, increasingly with a more coarsen grid, the field has sources and sinks using if the approximative model is used.

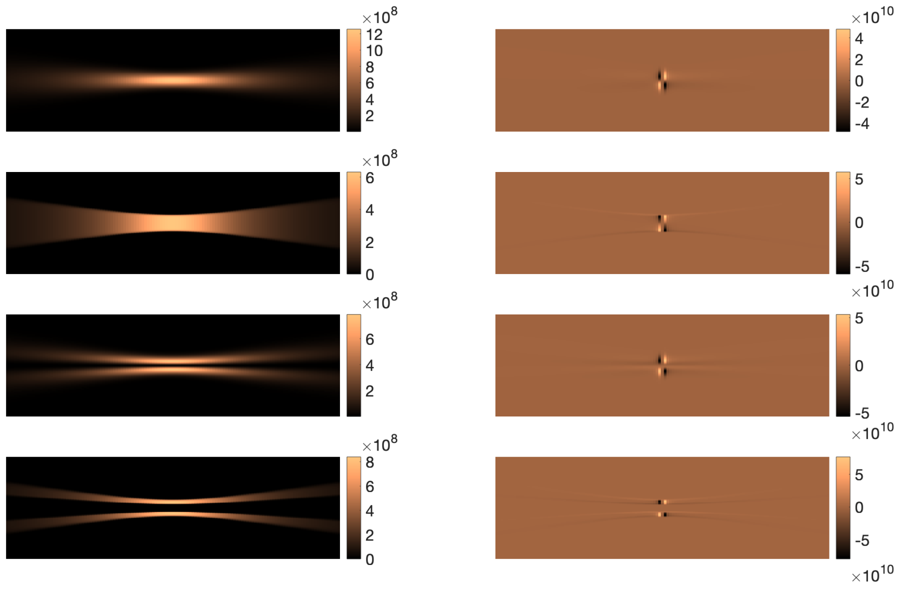


Figure 4.6.: Left: intensity $I(r, z)$ different beam profiles, right: divergence of the modeled intensity flux $\nabla(\vec{n}I(r, z))$ by equation (4.13), Mesh: 128x128 (generated in MATLAB R2019b, code (opticalPotential.m) appended in chapter A).

4.1. Unidirectional energy flux $\vec{n}_l$

To compute the energy transfer from a laser beam to the material in space, a direction for the flux of this transfer has to be found. Since this process is a field problem, the direction usually depends on the location of the field point. For the sake of simplicity, this function can be neglected as well as an approximation. This can be argued by the fact, that the energy flux direction of laser material processing can be considered as the optical axis for the whole field. That is especially fulfilled when the beam has a small divergence that is related to good beam quality. In macroscopic processes that is often a very good approximation.

4.1.1. Hexahedral cell projection

If hexahedrons are used for the cell of a finite volume discretization, the projection of those objects is of interest to calculate the flux through the surface of the object. This is done by the operator rayf in our model. The computation by Heron's formula can be done as well.

4.2. Trigonometrical, rotationally symmetric energy flux $\vec{n}^*$

The local energy flux can not be considered as unidirectional in some cases. For that reason the flux direction of the beam can be modeled trigonometrical, if rotational symmetry is assumed for the beam. The unit vector for the direction can be computed by,

$$\sin(\alpha(r,z)) = \frac{r}{z\left[1+\left(\frac{z_R}{z}\right)^2\right]}. \tag{4.14}$$

Using the Euler identity, we can state,

$$\cos(\alpha(r,z)) = \sqrt{1-\sin(\alpha(r,z))^2}. \tag{4.15}$$

This leads to the trigonometrical, rotationally symmetric energy flux,

$$\vec{n}^*(\vec{r}) = \cos(\alpha(r,z))\vec{e}_z + \sin(\alpha(r,z))\vec{e}_r. \tag{4.16}$$

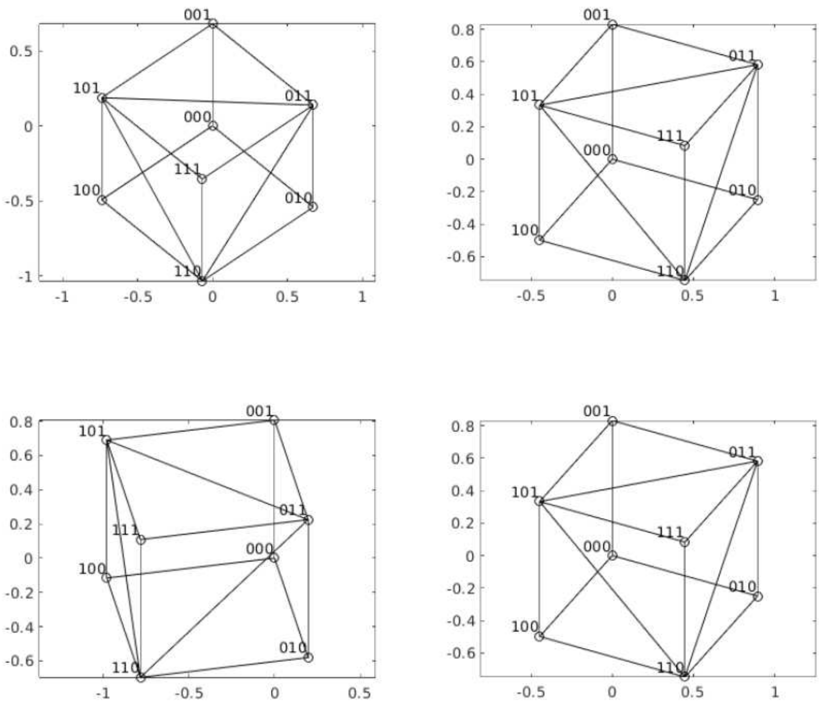


Figure 4.7.: A projected hexahedron from different angles, where the projected plane is normal to the energy flux direction and is assumed to be constant in a cell (generated in MATLAB R2019b, code (hexahedronProjection.m) appended in chapter A).

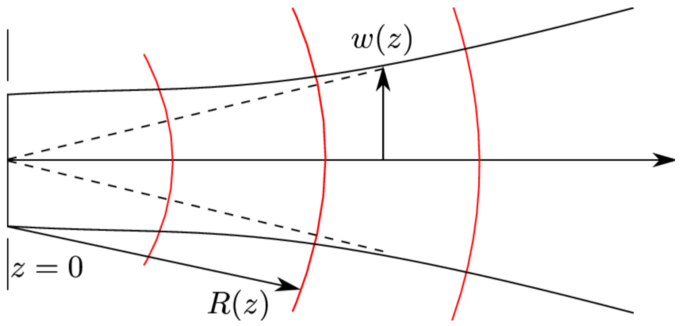


Figure 4.8.: Wave front curvature of the Gaussian beam R(z), used for the trigonometrical, rotationally symmetric energy flux direction

4.2.1. Gaussian beam caustic

In Figure 4.9 the beam caustic of a Gaussian beam can be seen, which is defined as a surface $f(r, \varphi, z) = \frac{r}{W(z)} - 1 = 0$ in polar coordinates. The color implies the normal distance to the beam origin. This surface was implemented for visualization reasons and as a helping field as criteria for subdividing integration regions in OpenFoam as well. This surface is equal to the Tophat beam regions border and could be used as numerical tool for identifying critical cells, which intersect with the border of the two different regions.

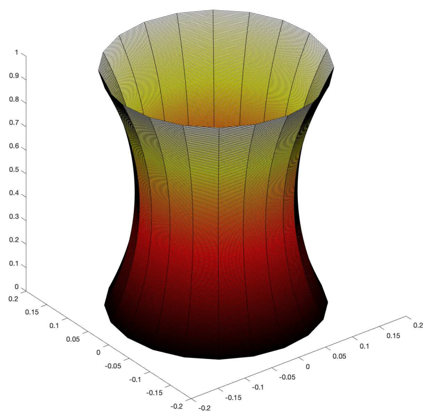


Figure 4.9.: Beam caustic of a Gaussian beam, z colored (generated in MATLAB R2019b, code (BeamShapes.m) appended in chapter A).

4.3. Potential theory energy flux

To couple the laser physics with the fluid dynamics in laser material processing, it is of relevance to know the energy flux direction of the quasi-steady solution of the optical field. Due to the necessity of much smaller cells for resolving wave functions of the beam, compared to the cell size of the underlying fluid dynamic problem, the energy transfer is generally modeled as a transport process of the intensity. Because we are mainly interested in beam shapes described in polar coordinates, we consider the solution for Laguerre-Gaussian [2] beams:

$$U_{l,m}(r,\theta,z) = A_{l,m}\Big[\frac{W_0}{W(z)}\Big]\Big(\frac{r}{W(z)}\Big)^l \mathbb{L}_m^l\Big(\frac{2r^2}{W^2(z)}\Big) \cdot \exp\Big(-\frac{r^2}{W^2(z)}\Big) \times \exp\Big(\Phi_{l,m}(r,\theta,z)\Big).$$

Whereby,

$$\begin{aligned} \Phi_{l,m}(\vec{r}) &= -j(kz+l\theta) - jk\frac{r^2}{2R(z)} + j(l+2m+1)\zeta(z), \\ \vec{r} &= (r,\theta,z)^T, \\ \mathbb{L}_m^l(x) &= (x^{-l}e^x/m!)(d^m/dx^m)(x^{l+m}e^{-x}), \\ W(z) &= W_0\sqrt{1+(\frac{z}{z_0})^2}, \\ R(z) &= z\Big[1+(\frac{z_0}{z})^2\Big], \\ \zeta(z) &= \arctan\frac{z}{z_0}, \\ z_0 &= \frac{\pi W_0^2}{\lambda \mathbb{M}^2}. \end{aligned} \tag{4.17}$$

Beams for laser material processing have typically a non-perfect beam quality, by this, the Rayleigh range has been scaled by a factor $1/\mathbb{M}^2$ in the model.
Due to the discrepancy between the time-step-resolution of the transient fluid and laser physics problem, we are interested in a model for the intensity field and its time-averaged flux direction. This can be understood as the quasi-steady solution for the Poynting vector field $\langle\vec{S}(\vec{r})\rangle$ of a laser beam. If the intensity profile $I(\vec{r})$ is given, the energy flux direction $n(\vec{r})$ has to be modeled separately.

$$\langle\vec{S}(\vec{r})\rangle = \vec{n}(\vec{r})||\langle\vec{S}(\vec{r})\rangle|| = \vec{n}(\vec{r})I(\vec{r}) \tag{4.18}$$

$$\langle\vec{S}(\vec{r})\rangle = \frac{1}{T}\int_0^T \vec{S}(\vec{r})\rangle dt$$

4.3.1. The phase-model $\vec{n}'$

A multi-mode intensity profile can be obtained by a superposition of their fundamental modes, like the Laguerre-Gaussian modes. Due to that we are interested in the time-averaged energy flux direction of those beams, which can be modeled by,

$$\vec{n}_{l,m}(r,\theta,z) = \frac{-\nabla\Im(\Phi_{l,m}(r,\theta,z))}{||\nabla\Im(\Phi_{l,m}(r,\theta,z))||}. \tag{4.19}$$

After assuming rotationally symmetric beams ($l = 0$) we can derive,

$$\nabla = \vec{e}_r \frac{\partial}{\partial r} + \frac{1}{r}\vec{e}_\theta \frac{\partial}{\partial \theta} + \vec{e}_z \frac{\partial}{\partial z}$$

$$-\nabla \Im(\Phi_{0,m}(r,z)) = \vec{e}_r \frac{krz}{z_0^2 + z^2} + \vec{e}_z \Bigg[\underbrace{(2m+1)\frac{-z_0}{z_0^2+z^2}}_{\text{Gouy phase term}} + \frac{k(2z_0^4 + z_0^2(r^2+4z^2) - r^2z^2 + 2z^4)}{2(z_0^2+z^2)^2} \Bigg]. \quad (4.20)$$

If we consider the Gouy phase term as neglectable [15], which is especially the case for $(2m+1) \ll k = 2\pi/\lambda$, we can conclude,

$$\nabla \Phi_{0,m}(r,z) \approx \nabla \Phi_{0,0}(r,z). \quad (4.21)$$

This leads to the analytic solution for a model of the energy flux direction of a rotationally symmetric beam of an arbitrary order or a superposition of those, as the flux direction of the fundamental Gaussian mode,

$$\vec{n}'(r,z) = \frac{1}{\xi(r,z)} \Bigg[\vec{e}_r \xi_r(r,z) + \vec{e}_z \xi_z(r,z) \Bigg],$$
$$\xi(r,z) = \sqrt{\xi_r^2(r,z) + \xi_z^2(r,z)},$$
$$\xi_r(r,z) = \frac{krz}{z_0^2+z^2},$$
$$\xi_z(r,z) = \frac{k(2z_0^4 + z_0^2(r^2+4z^2) - r^2z^2 + 2z^4)}{2(z_0^2+z^2)^2}. \quad (4.22)$$

The model for the energy flux direction $\vec{n}'(r,z)$ was implemented in OpenFoam, visualized in Paraview, and can be seem in figure 4.11. The introduced vector field will be used as the initial field when the later introduced optical potential is calculated. This can be argued if the time-averaged Poynting theorem or divergence theorem is used,

$$\nabla \langle \vec{S}(\vec{r}) \rangle = -\frac{1}{2}\omega_0 \epsilon_0 \epsilon''(\omega_0) U_{l,m}(\vec{r}) U_{l,m}^*(\vec{r}). \quad (4.23)$$

The term $\nabla \Phi_{0,0}(r,z)$ represents a vector that points into the direction of the biggest increase of the function Φ. The magnitude of this function is plotted in figure 4.10. The assumption for phase-model $\vec{n}'$ is, that the energy flux points into that direction.

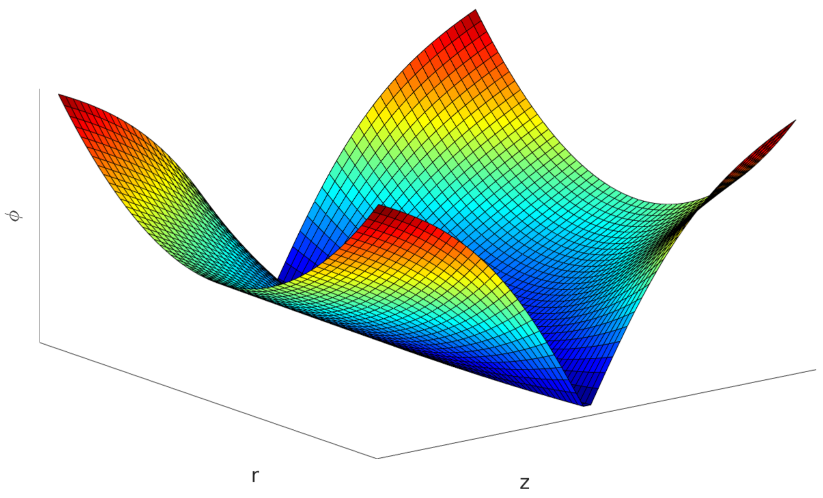


Figure 4.10.: $f : \phi = |\Phi_{0,0}(r, z)|$ (generated in MATLAB R2019b, code (opticalPotential.m) appended in chapter A).

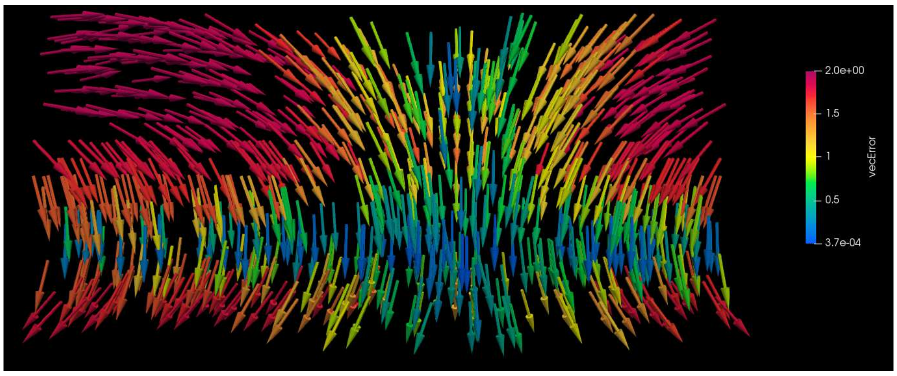


Figure 4.11.: Propagation direction of the modeled beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

4.3.2. The optical potential analogy

If the imaginary permittivity ϵ'' is zero, we know that $\langle\vec{S}(\vec{r})\rangle$ has to be solenoidal. As an analogy to fluid dynamics, where a scalar potential is introduced and it is assumed that φ can fulfill the basic law of fluid dynamics, we introduce a scalar potential Φ which can model the beam propagation, what implies that the modeled quasi-steady solution is a conservative vector field. This leads, like in fluid dynamics, to the Laplace's equation. Since our beam propagation model is implemented in a finite volume code, we are interested in the related Poisson's equation, [10]

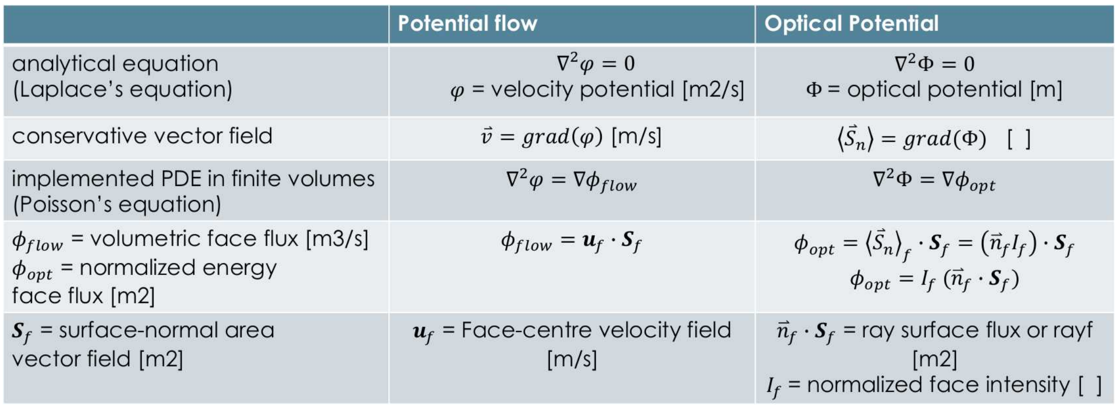
$$\nabla^2\Phi = \nabla\phi_o. \tag{4.24}$$

The energy surface flux through a face area is computed by,

$$\phi_o = \langle\vec{S}(\vec{r})\rangle_f \cdot \boldsymbol{S}_f,$$

$$\phi_o = \big(\vec{n}'_f(r,z)I_f(\vec{r})\big)\cdot\boldsymbol{S}_f = I_f(\vec{r})\big(\vec{n}'_f(r,z))\cdot\boldsymbol{S}_f\big). \tag{4.25}$$

The index f denotes that the field is interpolated between the two surrounding cells at a face. The surface area vector is denoted by $\boldsymbol{S}_f$ again.
This analogy can calculate the energy flux direction of the beam under the given assumptions. The reason for implementing the Poisson's equation has a mathematical origin. The definition of the Laplace equation exists only in the infinite small field point. Since we are doing numerical multi-physical simulations, we are limited to finite-sized grids. Consequently, we compute an equation, which gives the idea of a potential function, by computing a PDE which exists of a spatial second derivative term of the potential function on the one hand plus the divergence of the flux term through a closed finite volume. This equation is known as Poisson's equation. A solution for the numerical technique of that equation exists, if the initial field is already close to its final solution. For that reason, we need to couple the optical potential analogy with the n'-phase model.



	Potential flow	Optical Potential
analytical equation (Laplace's equation)	$\nabla^2\varphi = 0$ φ = velocity potential [m2/s]	$\nabla^2\Phi = 0$ Φ = optical potential [m]
conservative vector field	$\vec{v} = grad(\varphi)$ [m/s]	$\langle\vec{S}_n\rangle = grad(\Phi)$ [ ]
implemented PDE in finite volumes (Poisson's equation)	$\nabla^2\varphi = \nabla\phi_{flow}$	$\nabla^2\Phi = \nabla\phi_{opt}$
ϕ_{flow} = volumetric face flux [m3/s] ϕ_{opt} = normalized energy face flux [m2]	$\phi_{flow} = \boldsymbol{u}_f \cdot \boldsymbol{S}_f$	$\phi_{opt} = \langle\vec{S}_n\rangle_f \cdot \boldsymbol{S}_f = (\vec{n}_f I_f) \cdot \boldsymbol{S}_f$ $\phi_{opt} = I_f\,(\vec{n}_f \cdot \boldsymbol{S}_f)$
$\boldsymbol{S}_f$ = surface-normal area vector field [m2]	$\boldsymbol{u}_f$ = Face-centre velocity field [m/s]	$\vec{n}_f \cdot \boldsymbol{S}_f$ = ray surface flux or rayf [m2] I_f = normalized face intensity [ ]

Figure 4.12.: Overview of the analogy

In the overview of the analogy, the differences of the certain potential theories can be seen. Both theories have from a mathematical point of view in common, that the PDE would be the Laplace's equation analytically. Both solutions are computed in finite volumes, which leads to the Poisson equation in both cases. Their differences are the physical meanings of the quantities. On the one hand, a velocity potential is computed and on the other hand, an intensity or energy flux potential is computed. Consequently, the different fields have different certain units which are given in the overview.

5. Simulation results in OpenFoam

The initially analyzed laser beam can be seen in figure 5.1. The shown intensity field is a Gaussian beam, which was computed by the equation (4.6) in finite volumes. The volume-averaged intensity field was integrated by the cell area and averaged in this region. This was done by a polar integration method. The beam model was analyzed concerning its flux direction by different beam propagation models. Their flux directions are visualized in a later section. In figure 5.2 the results of the energy flux direction field, if a trigonometrical computed, rotationally symmetric beam propagation approach is used, can be seen. The result is rotationally symmetric and is equal to the potential theory approach at the beam origin's plane and along the optical axis. The color of the vectors gives the difference to the potential theory approach. Consequently, the vectors at the beam origin's plane and the beam axis are nearly zero. The red color and a value of 2.0 means, that the direction of the other approach is exactly in the opposite direction. It can be concluded, that the error of the trigonometrical computed, rotationally

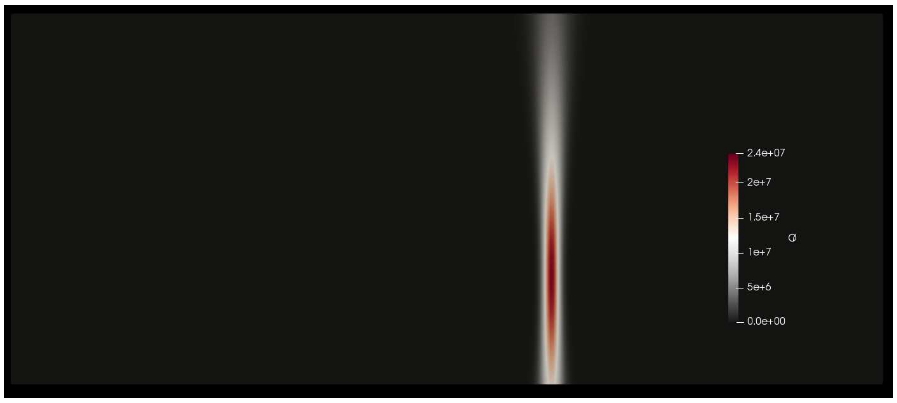


Figure 5.1.: Intensity field of the initial Gaussian beam which was used as a validation case (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

symmetric energy flux direction especially affects the intensity transport far away from the beam's origin. Typically, the intensity in those areas is much smaller than the intensity in the center of the beams field. As a result, the energy flux error is not that large in those distanced areas, even if the direction points in the opposite direction.

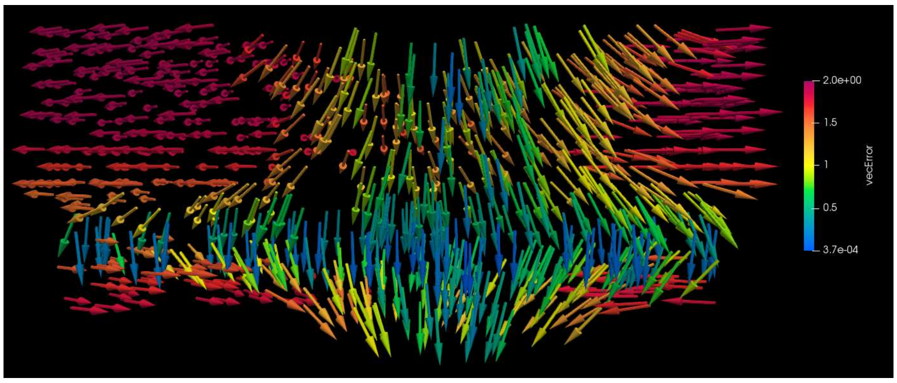


Figure 5.2.: Computed trigonometrical, rotationally symmetric energy flux $\vec{n}$ for the Gaussian beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

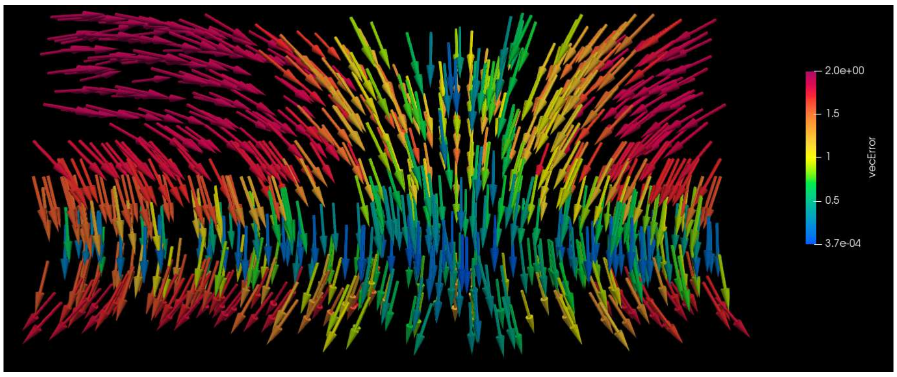


Figure 5.3.: Computed energy flux $\vec{n}$ for the Gaussian beam using the optical potential and the phase-n' model (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

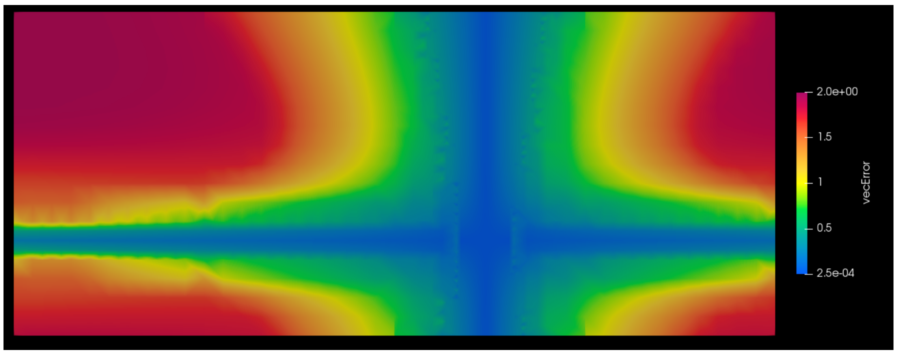


Figure 5.4.: Difference of the unidirectional energy flux direction vector and the trigonometrical, rotationally symmetric energy flux vector for a Gaussian beam $||\vec{n}_l - \vec{n}^*(\vec{r})||$ (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

In figure 5.5 - 5.15 the simulation results of the new beam propagation approach, which uses the n'-phase model and the optical potential analogy simultaneously, can be seen.

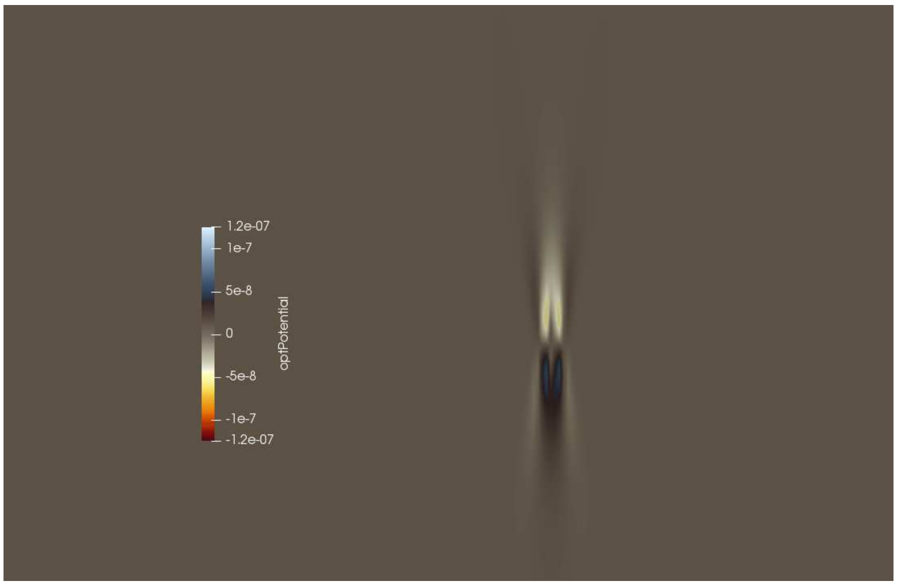


Figure 5.5.: Modeled beam potential Φ for a Gaussian beam (generated in Open-Foam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

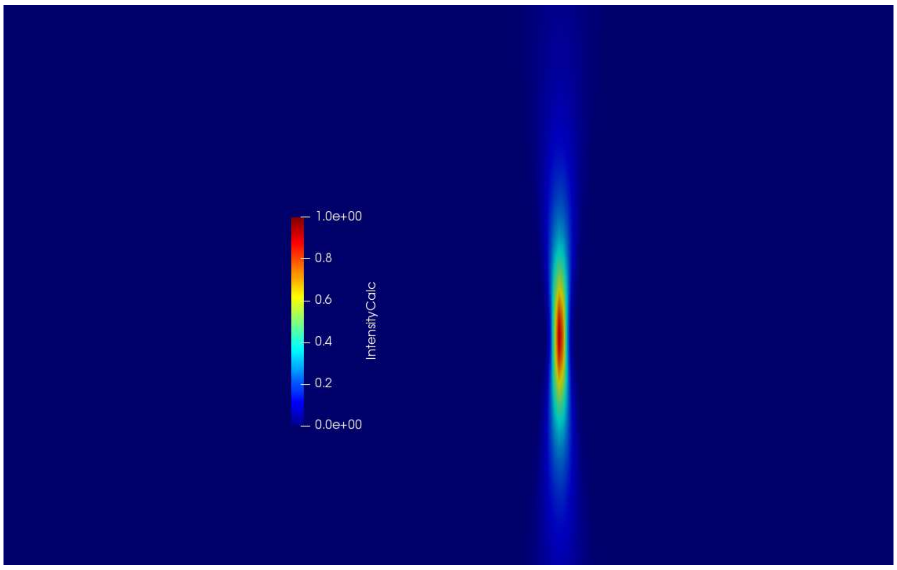


Figure 5.6.: Resulting intensity field $I_\Phi(\vec{r})$ for a Gaussian beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

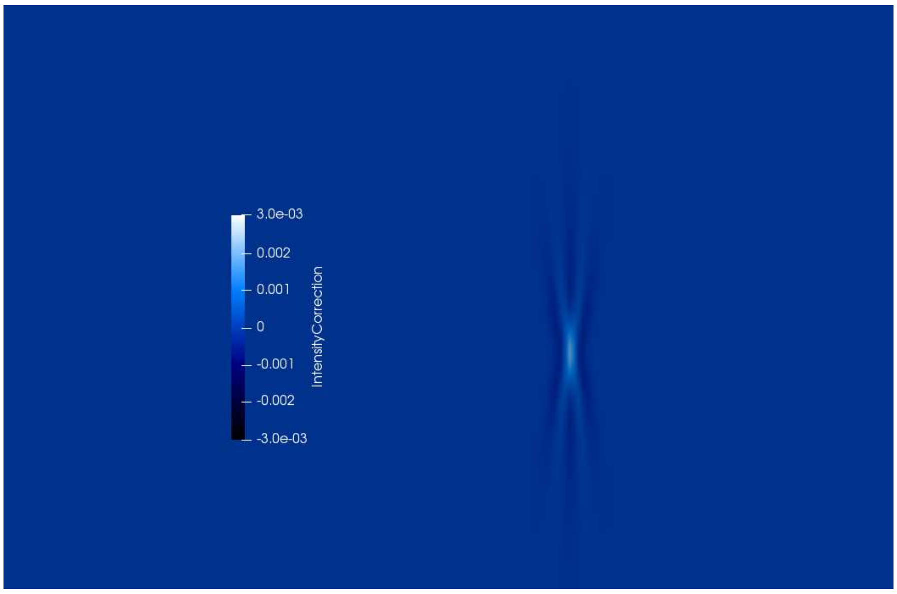


Figure 5.7.: Correction of the Gaussian intensity field $|I(\vec{r}) - I_\Phi(\vec{r})|$ (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

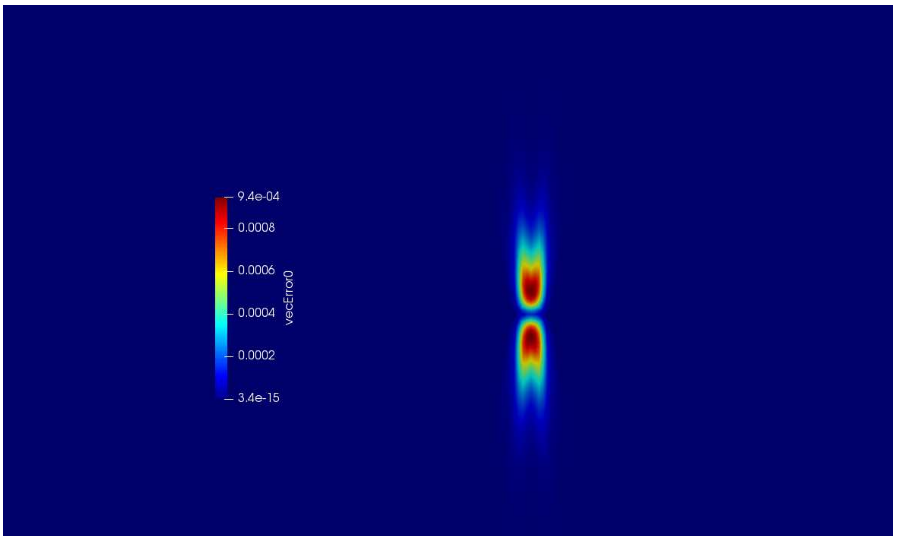


Figure 5.8.: Flux direction correction $||\vec{n}'(\vec{r}) - \vec{n}_{\Phi}(\vec{r})||$ of a Gaussian beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

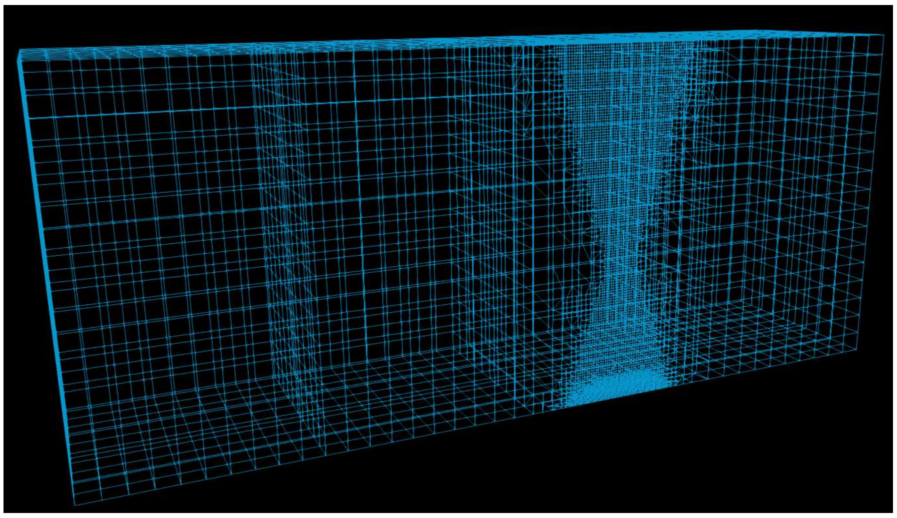


Figure 5.9.: Mesh geometry of the validation case (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

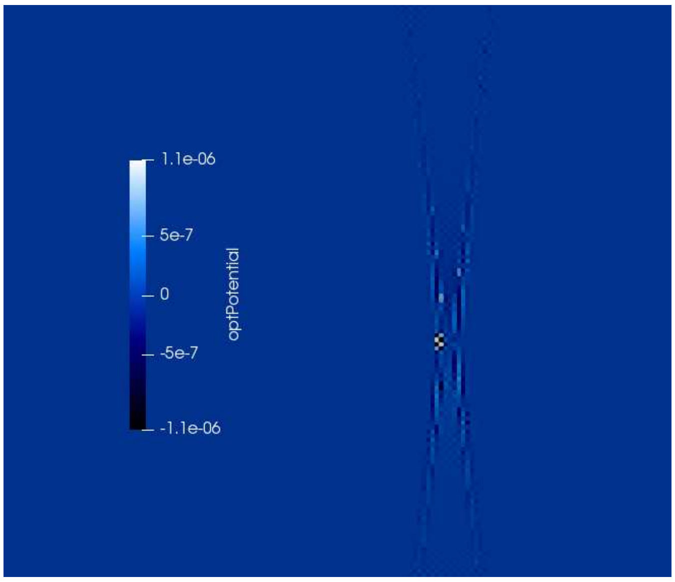


Figure 5.10.: Modeled optical potential Φ of a tophat beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

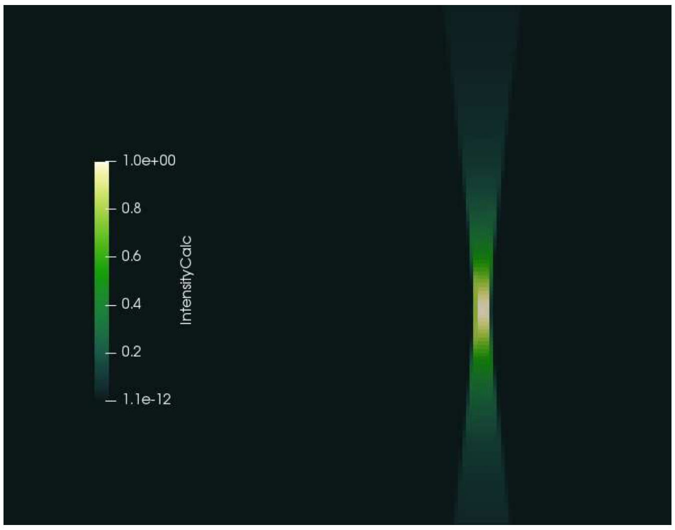


Figure 5.11.: Resulting intensity field $I_\Phi(\vec{r})$ of a tophat beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

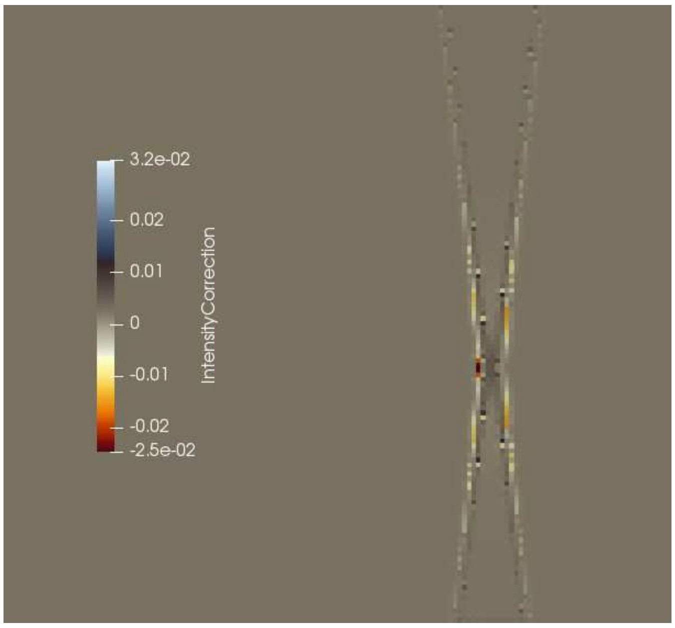


Figure 5.12.: Correction of the intensity field $|I(\vec{r}) - I_\Phi(\vec{r})|$ of a tophat beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

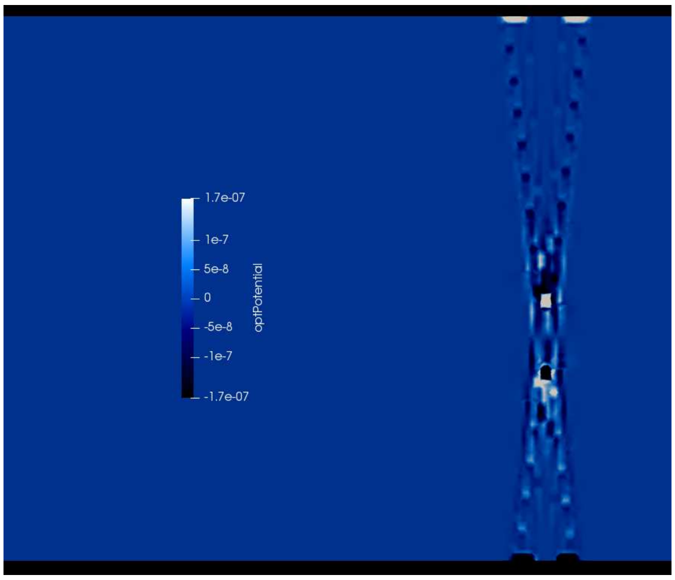


Figure 5.13.: Modeled beam potential Φ of a ring beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

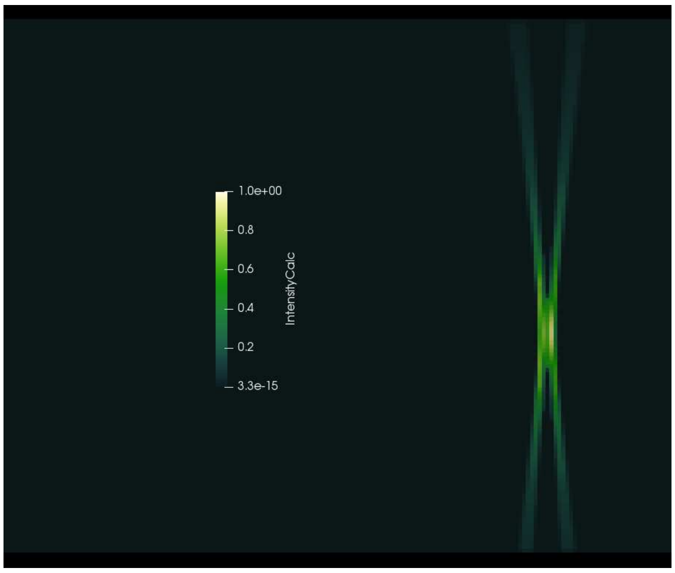


Figure 5.14.: Resulting intensity field $I_\Phi(\vec{r})$ of a ring beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

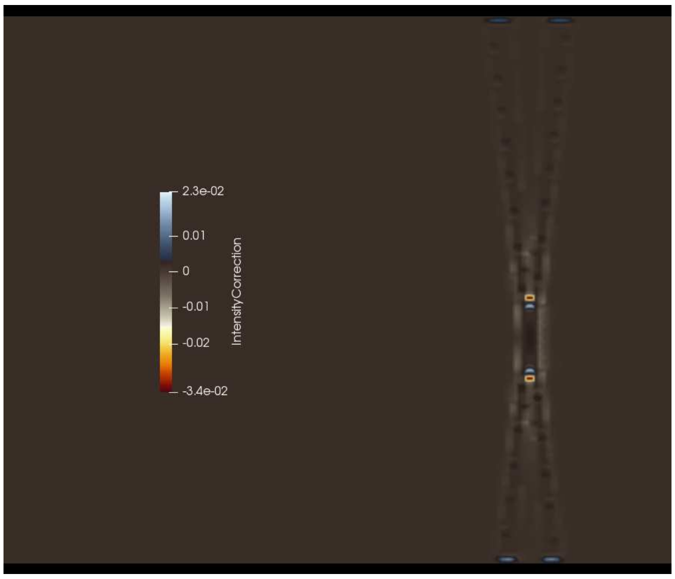


Figure 5.15.: Correction of the intensity field $|I(\vec{r}) - I_\Phi(\vec{r})|$ of a ring beam (generated in OpenFoam 6 and visualized by Paraview 5.6, code developed by IFT, Vienna University of Technology).

In the old approach, an approach wherein the energy flux in a field point was assumed to be unidirectional, the energy flux was computed by projecting the hexahedral cells on a plane, which is perpendicular to the optical axis. That approach was connected to the problem that, if for example a ring mode was not resolved well in the focus and the beam propagation after the focus point should be simulated, the shape of the beam could not be reconstructed in the diverging region again. The information on the beam shape was lost in the focus point. Using a converging-diverging flux model, similar to a flow problem in a converging-diverging nozzle, the beam shape can be reconstructed by knowing the direction of the energy flux.

Consequently, the approximative assumption of the unidirectional energy flux cannot be used anymore, the direction of the energy flux must be computed for each cell. Additionally, errors concerning energy conservation can be reduced. This comes by the fact, that the new energy flux model is based on the idea, that the intensity field of the laser beam multiplied by the computed energy flux direction forms a solenoidal vector field, what is ensured by the condition of the existence of a potential function of the vector field.

The new model was tested for a Gaussian beam in figure 5.5 - 5.8, for a tophat beam in figure 5.10 - 5.12, and a ring beam in figure 5.13 - 5.15.

The assumption of the unidirectional beam propagation is typically connected to small energy conservation errors if the beam diverges weakly for the size of the processing domain. For problems wherein the angle of the diverging paraxial beam is large, the assumption of a unidirectional energy flux can lead to energy conservation problems. This can be avoided by, if a local energy flux model, like presented in this work, is used. Figure 5.5 shows the optical potential, computed by implementing equation (4.24) in OpenFoam for a Gaussian beam with a beam diameter of 0.5mm and a $\mathbb{M}^2$ of 60. The laser power was set to 10W, the scaling of the result does not depend on that, since the initial intensity field is normalized to $I_{max} = 1$. The two other presented beam shapes were computed using the same beam parameters. Figure 5.6 shows the derived intensity field which could be obtained by using the beam potential of the Gaussian beam, shown in the figure above. The change of the initial intensity field and intensity derived by the potential can be seen in figure 5.7. Dirichlet boundaries were applied on the upper and lower boundary for the beam potential equation, $\Phi(y = 0) = \Phi(y = H) = 0$. The PDE was solved with the Gauss-Seidel algorithm. The change from initial to computed intensity can be seen in Figure 5.7. The flux direction correction is visualized in figure 5.8. The mesh of the finite volume code was refined by the intensity of the beam in our tests for all beams and can be seen in figure 5.9. Not only the Gaussian beam could be recomputed by the analogy, but even higher-order paraxial beams like the tophat beam could also be recomputed by the presented model.

The results for the beam potential are shown in figure 5.10, the derived intensity field in figure 5.11, and the final correction compared to the initial field in figure 5.12. The effect of reconstructing the beam shape, even if the beam profile is not resolved anymore in the focus point, can be seen in the result of the ring mode in figure 5.14. The potential of that beam is shown in figure 5.13. The correction in the following figure 5.15, wherein can be realized, that the main correction of the beam was done in the region of the ring mode field, where the ring shape began to be unresolved in the grid of the laser model.

6. Conclusion

In multi-physical simulations for laser material processing, the direction of the energy flux has to be considered to describe the transient process without infringing physical laws, like energy conservation. Errors can arise numerically or can be inherent in an approximative model. To understand this problem, different models for the radiation flux of a laser beam through the grid cells were discussed.
Previously, the physics background for the specific problem was given. The use of the models, the coupling between laser and fluid was introduced. An absorption model for the energy transfer from laser to fluid is necessary for that use, which has to be a numerical model, discretized in finite volumes.
The different models for the radiation flux, unidirectional, trigonometrical, rotationally symmetric, and computed by a potential theory were explained. The difference between the conservative and the non-conservative flux model is explained and different beam shapes, which are typically used in laser material processing, were analyzed. A possible conservative flux model is introduced as an analogy to potential flow theory in fluid dynamics. The equations for different beam shapes are implemented in Matlab (code appended) and their mathematical properties were analyzed numerically. Beam propagation approaches, which are directly related to the energy flux model in finite volumes, are implemented and tested in OpenFoam. The obtained numerical solutions were visualized in Paraview. The possibility of computing laser beams by the Laplace equation, analog to the computation to the velocity field in fluid dynamics, was shown.

Bibliography

[1] A. Andersson et al. *Computational Fluid Dynamics for Engineers*. Cambridge University Press, (2012).

[2] M.C. Teich B.E.A. Saleh. *Fundamentals of Photonics. Second Edition*. Wiley Series in Pure and Applied Optics, (2007).

[3] F.-J. Gürtler et al. *Simulation of Laser Beam Melting of Steel Powders using the Three-Dimensional Volume of Fluid Method*. Physics Procedia, 41, 881–886., (2013).

[4] S. A. Khairallah and A. Anderson. *Mesoscopic simulation model of selective laser melting of stainless steel powder*. Journal of Materials Processing Technology, 214(11), 2627–2636., (2014).

[5] F. Kong and R. Kovacevic. *Modeling of Heat Transfer and Fluid Flow in the Laser Multilayered Cladding Process*. Metallurgical and Materials Transactions B, 41(6), 1310–1320., (2010).

[6] Y. S. Lee and W. Zhang. *Modeling of heat transfer, fluid flow and solidification microstructure of nickel-base superalloy fabricated by laser powder bed fusion*. Additive Manufacturing, 12, 178–188., (2016).

[7] T. G. Mayerhöfer and J. Popp. *Beer's law derived from electromagnetic theory*. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, (2019).

[8] J. Mazumder. *Overview of melt dynamics in laser processing*. Optical Engineering, 30(8), 1208., (1991).

[9] A. Minachi et al. *Predictions of the Gauss-Hermite beam model and finite element methodf or ultrasonic propagation through anisotropic stainless steel*. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, 40(4), 338–346., (1993).

[10] B. Nebenfuhr. *OpenFOAM: A tool for predicting automotive relevant flow fields*. Department of Applied Mechanics, Chalmers University of Technology, (2010).

[11] A. Otto, H. Koch, and R. G. Vazquez. *Multiphysical Simulation of Laser Material Processing*. Physics Procedia, 39, 843–852., (2012).

[12] A. Otto et al. *Numerical Simulations - A Versatile Approach for Better Understanding Dynamics in Laser Material Processing*. Physics Procedia 12 11–20, (2011).

[13] H. Qi, J. Mazumder, and H. Ki. *Numerical simulation of heat transfer and fluid flow in coaxial laser cladding process for direct metal deposition*. Journal of Applied Physics, 100(2), 024903., (2006).

[14] G. Taylor et al. *Finite volume methods applied to the computational modelling of welding phenomena*. Applied Mathematical Modelling 26 309–320, (2002).

[15] X. Wang et al. *Complete presentation of the Gouy phase shift with the THz digital holography*. Optics Express, 21(2), 2337., (2013).

A. Matlab-code

opticalPotential.m

```
clear all
clc
close all

P     = 1000;          %Watt
lambda = 1064*1e-9;
Wo    = 1.3*lambda;%1e-3;   %meter
discret = 557;%247-0*220-0*230;
k = 2*pi/lambda;
M2 = 1;

underrelax=0.3;

pfactor=1;

zo    = pi*Wo^2/lambda;
Y = 12.0*lambda;
%Z = 1.0*zo;
Z=Y/2;


z =  [-Z:(Z/discret*2):Z].*ones(discret+1);
r = -[-Y:(Y/discret*2):Y]'.*ones(discret+1);

I_gauss=0.0*r;
I_ring=0.0*r;
I_supergauss=0.0*r;
I_TM01=0.0*r;


for m = 1:length(r(:,1))
      for j = 1:length(r(1,:))

            W       = Wo*sqrt(1+(z(m,j)/zo*M2)^2);
            R_ax    = z(m,j)*(1+(zo/z(m,j))^2);
            W2      = W/2;
            I_gauss(m,j)=(2*P/(pi*W^2))*exp(-2*r(m,j)^2/W^2);
            I_TM01(m,j)=(2*P/(pi*(W^2-W2^2)))*(exp(-2*r(m,j)^2/W^2)
                ...
                                    -exp(-2*r(m,j)^2/W2^2));
            I_supergauss(m,j)=(P/(pi*W^2))*exp(-2*r(m,j)^16/W^16);
            I_ring(m,j)=(P/(pi*(W^2-W2^2)))*(exp(-2*r(m,j)^16/W^16)
                ...
                                    -exp(-2*r(m,j)^16/W2^16));

            %A(i,j)=(A0/z(i,j)*exp(-i*k*r(i,j)^2/2/z(i,j)));
            U(m,j) = Wo/W*exp(-r(m,j)^2/W^2)*exp(-i*k*z(m,j)-i*k*r(m
                ,j)^2/2/R_ax+i*atan(z(m,j)/zo));

            front_z(m,j)=cos(asin(r(m,j)*1000/R_ax));
            front_r(m,j)=sin(asin(r(m,j)*1000/R_ax));

            x_p = z(m,j);
            r_p = r(m,j);

            R = x_p*(1+(zo/x_p)^2);

            for e =1:10

                x_r = -x_p + R - sqrt(R^2-r_p^2);

                R = x_p*(1+(zo/x_p)^2);

                R_plane(m,j) = R;

            end

            phase(m,j) = k*z(m,j) + k*r(m,j)^2/(2*R_ax)  - atan(z(m,
                j)/zo);

        end
end


[phase_gradX,phase_gradY]=gradient(phase);
%phase = sqrt(phase_gradX.^2+phase_gradY.^2);

phase_gradX=phase_gradX./sqrt(phase_gradX.^2+phase_gradY.^2);
phase_gradY=phase_gradY./sqrt(phase_gradX.^2+phase_gradY.^2);


I_gauss=I_gauss/max(max(I_gauss));
%U = U/max(max(abs(U)));
I_TM01=I_TM01/max(max(I_TM01));
I_supergauss=I_supergauss/max(max(I_supergauss));
```

```
I_ring=I_ring/max(max(I_ring));


for i = 1:length(r(:,1))-1
    for j = 1:length(r(1,:))-1

        Nabla_x(i,j)=(I_gauss(i+1,j)-I_gauss(i,j))/(Z/discret)
            ...
             +(I_gauss(i,j+1)-I_gauss(i,j))/(Y/discret);

    end
end
%I_gauss=I_ring;
%R=0.0*r;


% for i = 1:length(r(:,1))
%     for j = 1:length(r(1,:))
%
%         R(i,j)=z(i,j);%*(1+(zo/z(i,j))^2);
%
%     end
% end
%
% RN = sqrt(R.^2+r.^2);

[gradIx,gradIy]=gradient(sqrt(I_gauss));


phi1=-gradIx./sqrt(gradIx.^2+gradIy.^2);
phi2=-gradIy./sqrt(gradIx.^2+gradIy.^2);



% phi1=front_z;
% phi2=front_r;



figure(4)
contourf(phi1)

%phi1 =  R./RN;
%phi2 =   r./RN*0.0;

%phi2 = R./R;
%phi1 = 0.0*R;

```

```
%Nabla_phi=0.0*r;

% for i = 1:length(r(:,1))-1
%       for j = 1:length(r(1,:))-1
%
%             Nabla_z(i,j)=((1*I_gauss(i+1,j)-1*I_gauss(i,j))/(Z/
    discret)...
%                   +(0*I_gauss(i,j+1)-0*I_gauss(i,j))/(Y/discret));
%
%       end
% end

[IdX,IdY]=gradient(sqrt(I_gauss));
norm_gradI = sqrt(IdX.^2+IdY.^2);
[IdXX,XXX]=gradient(IdX);
[XXX,IdYY]=gradient(IdY);
laplace_I = IdXX+IdYY;
div_nI = (I_gauss.*laplace_I+IdX.*IdX+IdY.*IdY)./norm_gradI;
Nabla_gauss=div_nI;


for i = 1:length(r(:,1))-1
    for j = 1:length(r(1,:))-1

        %Nabla_gauss(i,j)=(phi1(i+1,j)*I_gauss(i+1,j)-phi1(i,j)*
            I_gauss(i,j))/(Z/discret)...
         %    +(phi2(i,j+1)*I_gauss(i,j+1)-phi2(i,j)*I_gauss(i,j)
            )/(Y/discret);
        %Nabla_gauss(i,j)=((phi1(i+1,j)-phi1(i,j))*I_gauss(i,j)+
            phi1(i,j)*(I_gauss(i+1,j)-I_gauss(i,j)))/(Z/discret)
            ...
          %                +0*((phi2(i,j+1)-phi2(i,j))*I_gauss(i,j
              )+phi2(i,j)*(I_gauss(i,j+1)-I_gauss(i,j)))/(Y/
              discret);

        %Nabla_gauss(i,j)=I_gauss(i,j)*((phi1(i+1,j)-phi1(i,j))
            /(Z/discret)...
         %    +0.0*(phi2(i,j+1)-phi2(i,j))/(Y/discret));


        %Nabla_gauss(i,j)=Nabla_gauss(i,j)/I_gauss(i,j);


        Nabla_supergauss(i,j)=(phi1(i+1,j)*I_supergauss(i+1,j)-
            phi1(i,j)*I_supergauss(i,j))/(Z/discret)...
              +(phi2(i,j+1)*I_supergauss(i,j+1)-phi2(i,j)*
                I_supergauss(i,j))/(Y/discret);
            Nabla_TM01(i,j)=(phi1(i+1,j)*I_TM01(i+1,j)-phi1(i,j)*
                I_TM01(i,j))/(Z/discret)...
                  +(phi2(i,j+1)*I_TM01(i,j+1)-phi2(i,j)*I_TM01(i,j))/(
                      Y/discret);
            Nabla_ring(i,j)=(phi1(i+1,j)*I_ring(i+1,j)-phi1(i,j)*
                I_ring(i,j))/(Z/discret)...
                  +(phi2(i,j+1)*I_ring(i,j+1)-phi2(i,j)*I_ring(i,j))/(
                      Y/discret);
      end
end

%div_0=div_0/max(max(div_0));

%Nabla_gauss=Nabla_gauss/max(max(Nabla_gauss));


% figure(1)
% subplot(1,2,2)
% set(gcf,'color','[1 1 1]');
% imagesc(I_gauss)
% %c=colorbar('southoutside');
% set(gca,'FontSize',20)
% %title('a=1')
% axis('equal')
% axis('off')
% grid('off')
% colormap(hot)

%W2    = sqrt(1+(r/zo).^2)';
%W3    = sqrt(1+(r/zo).^2);
%I_gauss2 = (2*P/(pi*W3.^2));%*exp(-2*(r.^2)./W2.^2);

figure(1)
subplot(4,3,1)
set(gcf,'color','[1 1 1]');
imagesc(I_gauss)
%imagesc(I_supergauss)
c=colorbar('eastoutside');
caxis([-0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
colormap(copper)
subplot(4,3,4)
set(gcf,'color','[1 1 1]');
imagesc(I_supergauss)
```

```
%imagesc(I_supergauss)
c=colorbar('eastoutside');
%caxis([-0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,7)
set(gcf,'color','[1 1 1]');
imagesc(I_TM01)
%imagesc(I_supergauss)
c=colorbar('eastoutside');
%caxis([-0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,10)
set(gcf,'color','[1 1 1]');
imagesc(I_ring)
%imagesc(I_supergauss)
c=colorbar('eastoutside');
%caxis([-0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')

subplot(4,3,2)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_gauss)
%caxis([-1 1])
%imagesc(R)
c=colorbar('eastoutside');
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
%colormap(jet)
subplot(4,3,5)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_supergauss)
%imagesc(R)
c=colorbar('eastoutside');
```

```
%caxis([-40 40])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,8)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_TM01)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-20 20])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,11)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_ring)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-60 60])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')


R=z.^2+r.^2;
%figure(4)
%quiver(x,y,phi1,phi2)

%subplot(2,1,2)
% figure(2)
% set(gcf,'color','[1 1 1]');
% imagesc(R)
% c=colorbar('eastoutside');
% set(gca,'FontSize',20)
% %title('a=1')
% %axis('equal')
% axis('off')
% grid('off')
% colormap(jet)

figure(3)
subplot(1,3,2)
```

```
set(gcf,'color','[1 1 1]');
imagesc(Nabla_gauss/max(max(Nabla_gauss)))
caxis([-1 1])
%imagesc(R)
c=colorbar('eastoutside');
%caxis([0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')


for k=1:2

      if k~=1
           Nabla_gauss=Nabla_gauss2;
      end

%Specifying parameters
nx=discret;                                 %Number of steps in space(
    x)
ny=discret;                                 %Number of steps in space(
    y)
niter=1;                            %Number of iterations
dx=2/(nx-1);                        %Width of space step(x)
dy=2/(ny-1);                        %Width of space step(y)
x=0:dx:2;                           %Range of x(0,2) and specifying
      the grid points
y=0:dy:2;                           %Range of y(0,2) and specifying
      the grid points
b=zeros(nx,ny);                     %Preallocating b
pn=zeros(nx,ny);                    %Preallocating pn

% Initial Conditions
if(k==1)
p=zeros(nx,ny);                     %Preallocating p
end

%Boundary conditions
% p(:,1)=0;
% p(:,ny)=0;
% p(1,:)=0;
% p(nx,:)=0;


i=2:nx-1;
j=2:ny-1;
```

```
%Explicit iterative scheme with C.D in space (5-point difference
    )
for it=1:niter
      pn=p;
      p(i,j)=((dy^2*(pn(i+1,j)+pn(i-1,j)))+(dx^2*(pn(i,j+1)+pn(i,j
          -1)))+(Nabla_gauss(i,j)*dx^2*dy*2))/(2*(dx^2+dy^2));
      %Boundary conditions
      p(:,1)=0;
      p(:,ny)=0;
      p(1,:)=0;
      p(nx,:)=0;


figure(2)
set(gcf,'color','[1 1 1]');
imagesc(p)
c=colorbar('eastoutside');
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
colormap(jet)
end

if(mod(k,2)==0)
      sign=1;
else
      sign=1;
end

[FX,FY] = gradient(sign*pfactor*p);

% figure(3)
% I_error = sqrt(FX.^2+FY.^2);
%
% contour(I_error)
% hold on
% quiver(FX,FY)
% hold off
% axis('off')
% grid('off')

% figure(4)
% imagesc(I_error)
% set(gcf,'color','[1 1 1]');
% c=colorbar('eastoutside');
% set(gca,'FontSize',20)
% axis('off')
% grid('off')
% colormap(jet)


for i = 1:length(phi1(:,1))-1
    for j = 1:length(phi2(1,:))-1

        FXn(i,j)=FX(i,j)/sqrt(FX(i,j)^2+FY(i,j)^2);
        FYn(i,j)=FY(i,j)/sqrt(FX(i,j)^2+FY(i,j)^2);

        %Intensity_error(i,j) = I_gauss(i,j)*Nabla_gauss(i,j)/
            max(max(Nabla_gauss));

        phi1raw(i,j)=(phi1(i,j)*(1-underrelax)+underrelax*FXn(i,
            j));
        phi2raw(i,j)=(phi2(i,j)*(1-underrelax)+underrelax*FYn(i,
            j));


        phi1(i,j)=phi1raw(i,j)/sqrt(phi1raw(i,j)^2+phi2raw(i,j)
            ^2);
        phi2(i,j)=phi2raw(i,j)/sqrt(phi1raw(i,j)^2+phi2raw(i,j)
            ^2);

    end
end



for i = 1:length(r(:,1))-1
    for j = 1:length(r(1,:))-1

        %Nabla_gauss2(i,j)=I_gauss(i,j)*((phi1(i+1,j)-phi1(i,j))
            /(Z/discret)...
        %     +(phi2(i,j+1)-phi2(i,j))/(Y/discret));

        Nabla_gauss2(i,j)=((phi1(i+1,j)-phi1(i,j))*I_gauss(i,j)+
            phi1(i,j)*(I_gauss(i+1,j)-I_gauss(i,j)))/(Z/discret)
            ...
                         +((phi2(i,j+1)-phi2(i,j))*I_gauss(i,j)+
                             phi2(i,j)*(I_gauss(i,j+1)-I_gauss(i,j
                             )))/(Y/discret);

        %Nabla_gauss2(i,j)=Nabla_gauss2(i,j)/I_gauss(i,j);

        Nabla_supergauss2(i,j)=((phi1(i+1,j)*I_supergauss(i+1,j)
            -phi1(i,j)*I_supergauss(i,j))/(Z/discret)...
```

```
                    +0.0*(phi2(i,j+1)*I_supergauss(i,j+1)-phi2(i,j)*
                        I_supergauss(i,j))/(Y/discret));
                Nabla_TM012(i,j)=((phi1(i+1,j)*I_TM01(i+1,j)-phi1(i,j)*
                    I_TM01(i,j))/(Z/discret)...
                    +0.0*(phi2(i,j+1)*I_TM01(i,j+1)-phi2(i,j)*I_TM01(i,j
                        ))/(Y/discret));
                Nabla_ring2(i,j)=((phi1(i+1,j)*I_ring(i+1,j)-phi1(i,j)*
                    I_ring(i,j))/(Z/discret)...
                    +0.0*(phi2(i,j+1)*I_ring(i,j+1)-phi2(i,j)*I_ring(i,j
                        ))/(Y/discret));
    end
end

%Nabla_gauss2=Nabla_gauss2/max(max(Nabla_gauss2));




figure(1)
subplot(4,3,3)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_gauss2)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-1 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
%colormap(jet)
subplot(4,3,6)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_supergauss2)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-40 40])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,9)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_TM012)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-20 20])
```

```
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
subplot(4,3,12)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_ring2)
%imagesc(R)
c=colorbar('eastoutside');
%caxis([-60 60])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')

%figure(6)
%imagesc(Intensity_error)


figure(3)

subplot(1,3,1)
set(gcf,'color','[1 1 1]');
imagesc(I_gauss)
%imagesc(I_supergauss)
c=colorbar('eastoutside');
%caxis([-0 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
colormap(copper)


subplot(1,3,3)
set(gcf,'color','[1 1 1]');
imagesc(Nabla_gauss2/max(max(Nabla_gauss2)))
%imagesc(R)
c=colorbar('eastoutside');
caxis([-1 1])
set(gca,'FontSize',20)
%title('a=1')
%axis('equal')
axis('off')
grid('off')
```

```
colormap(jet)

end

figure(5)

% subplot(1,3,1)
% FX0=FX./FX;
% FY0=0.0*FY;
%
% %grid('off')
% quiver(FX0,FY0)
% axis('off')
% set(gca,'XLim',[0.05*discret 0.95*discret],'YLim',[0.05*
    discret 0.95*discret])
%
[gradIx,gradIy]=gradient(sqrt(I_gauss));
FXN=-gradIx./sqrt(gradIx.^2+gradIy.^2);
FYN=-gradIy./sqrt(gradIx.^2+gradIy.^2);

% for i = 1:length(r(:,1))-1
%     for j = 1:length(r(1,:))-1
%     if(z(i,j)<0)
%         FXN(i,j)=-1*FXN(i,j);
%         FYN(i,j)=-1*FYN(i,j);
%     end
%     end
% end

subplot(1,2,1)
%axis('off')
%grid('off')
quiver(FXN,FYN)
axis('off')
set(gca,'XLim',[0.05*discret 0.95*discret],'YLim',[0.05*discret
    0.95*discret])

subplot(1,2,2)
%axis('off')
%grid('off')
quiver(phi1,phi2)
axis('off')
set(gca,'XLim',[0.05*discret 0.95*discret],'YLim',[0.05*discret
    0.95*discret])

figure(6)
quiver(phi1-FXN,phi2-FYN)
axis('off')
```

```
set(gca,'XLim',[0.05*discret 0.95*discret],'YLim',[0.05*discret
    0.95*discret])

figure(7)
imagesc(R_plane)
c=colorbar('eastoutside');
colormap(jet)
caxis([-60 60])

figure(8)
imagesc(imag(U))
%c=colorbar('eastoutside');
%colormap(winter)
xlabel("z")
ylabel("r")
set(gca,'xtick',[])
set(gca,'ytick',[])
set(gca,'FontSize',20)

figure(9)
surf(abs(phase))
grid('off')
xlabel("z")
ylabel("r")
zlabel("\phi")
set(gca,'xtick',[])
set(gca,'ytick',[])
set(gca,'ztick',[])
set(gca,'FontSize',20)
%caxis([0 100])
%c=colorbar('eastoutside');
colormap(jet)

% figure(10)
% quiver(phase_gradX,phase_gradY)
% %imagesc(phase)
% c=colorbar('eastoutside');
```

TEMmodes.m

```
clear all
clc
close all

P      = 1000;          %Watt
Wo      = 1.0*1e-4;   %meter
Wo3    = 0.6*1e-4;   %meter
lambda = 1064*1e-9;
```

```
discret = 173;%247-0*220-0*230;
k = 2*pi/lambda;
M2 = 60;

underrelax=0.3;

pfactor=1;

zo  = pi*Wo^2/lambda;
% Y = 50000.0*Wo;
% Z = 1.5*zo;
%Z=12*Y;

Y = 10*Wo;
Z = Y;

z =  [-Z:(Z/discret*2):Z].*ones(discret+1);
r = -[-Y:(Y/discret*2):Y]'.*ones(discret+1);


for m = 1:length(r(:,1))
    for j = 1:length(r(1,:))

            W      = Wo*sqrt(1+(z(m,j)/zo*M2)^2);
            W3       = Wo3*sqrt(1+(z(m,j)/zo*M2)^2);
            R_ax   = z(m,j)*(1+(zo/z(m,j))^2);

            I_gauss(m,j)=(2*P/(pi*W^2))*exp(-2*r(m,j)^2/W^2);

            I_TEM(m,j)=(2*P/(pi*W^2))*exp(-2*(sqrt(sqrt(r(m,j)^2*W
                ^2))-W)^2/W3^2);

            U(m,j)  = Wo/W*exp(-r(m,j)^2/W^2)*exp(-i*k*z(m,j)-i*k*r(m
                ,j)^2/2/R_ax+i*atan(z(m,j)/zo));

            wave(m,j)=real(U(m,j));


            x_p = z(m,j);
            r_p = r(m,j);

            R = x_p*(1+(zo/x_p)^2);

            for e =1:1

                x_r = -x_p + R - sqrt(R^2-r_p^2);

                R = x_p*(1+(zo/x_p)^2);
```

```

        end

        R_plane(m,j) = sqrt(R_ax^2+r(m,j)^2);

    end
end

theshold = max(max(abs(wave)))/10;

for m = 1:length(r(:,1))
    for j = 1:length(r(1,:))


        if(abs(wave(m,j))<theshold)
            wave(m,j)=0;
        end

                if(wave(m,j)<0)
            wave(m,j)=0;
        else
            wave(m,j)=1;
        end



    end
end

%[gradIx,gradIy]=gradient(R_plane);
%FXN=-gradIx./sqrt(gradIx.^2+gradIy.^2);
%FYN=-gradIy./sqrt(gradIx.^2+gradIy.^2);

FXN = cos(asin(r./R_ax));
FYN = -sin(asin(r./R_ax));
FXN = FXN./sqrt(FXN.^2+FYN.^2);
FYN = FYN./sqrt(FXN.^2+FYN.^2);


for m = 1:length(r(:,1))
    for j = 1:length(r(1,:))
        if(z(m,j)<0)
        FYN(m,j)=-FYN(m,j);
        end
    end
end


```

```
figure(1)
subplot(2,1,1)
% quiver(FXN,FYN)
% c=colorbar('eastoutside');
% colormap(jet)
% caxis([-60 60])
% subplot(3,1,2)
imagesc(I_TEM)
subplot(2,1,2)
imagesc(I_gauss)
% colormap(hot)
% c=colorbar('eastoutside');
% caxis([0 60])
% subplot(3,1,3)
% imagesc(I_gauss)
%
%
% figure(2)
% quiver(FXN,FYN)
```

hexahedronProjection.m

```
clc
clear all
close all

dx=1;  % x-dim
dy=1;  % y-dim
dz=1;  % z-dim
%dir = [1;2;1.5];
m = 1;
d = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ai=0*[1:m];
Ad=0*[1:d];
phi=0*[1:d];

SubPlots = 4;

for i = 1:m
for j = 1:SubPlots
if m > 1
dir = [1-2*rand(1,1);1-2*rand(1,1);1-2*rand(1,1)];
end
if j == 1 && m == 1
dir = [1.1;1;1.6];
end
if j == 2 && m == 1
```

```
dir = [1;2;1.5];
end
if j == 3 && m == 1
dir = [2;0.4;1.5];
end
if j == 4 && m == 1
dir = [1;2;1.5];
end

n1 = [1;0;0];
n2 = [0;1;0];
n3 = [0;0;1];
small = 0.0000000001*[1;1;1];
dir = dir + small;
dir = dir/norm(dir);

ortho1 = cross(dir,n3);
ortho1 = ortho1/norm(ortho1);
ortho2 = cross(ortho1,dir);
ortho2 = ortho2/norm(ortho2);

a_rho  = dot(n1*dy,ortho1);
a_zeta = dot(n1*dy,ortho2);
a      = sqrt(a_zeta^2+a_rho^2);

b_rho  = dot(n2*dx,ortho1);
b_zeta = dot(n2*dx,ortho2);
b      = sqrt(b_zeta^2+b_rho^2);

c_rho  = dot(n3*dz,ortho1);
c_zeta = dot(n3*dz,ortho2);
c      = sqrt(c_zeta^2+c_rho^2);

d1_v = -dy*n2+dz*n3;
d2_v =  dx*n1-dz*n3;
d3_v = -dx*n1+dy*n2;

d1_rho  = dot(d1_v,ortho1);
d1_zeta = dot(d1_v,ortho2);
d1      = sqrt(d1_rho^2+d1_zeta^2);

d2_rho  = dot(d2_v,ortho1);
d2_zeta = dot(d2_v,ortho2);
d2      = sqrt(d2_rho^2+d2_zeta^2);

d3_rho  = dot(d3_v,ortho1);
d3_zeta = dot(d3_v,ortho2);
d3 = sqrt(d3_rho^2+d3_zeta^2);
```

```

s1=(d1+b+c)/2;
s2=(d2+a+c)/2;
s3=(d3+a+b)/2;

A1=sqrt(s1*(s1-d1)*(s1-b)*(s1-c));
A2=sqrt(s2*(s2-d2)*(s2-a)*(s2-c));
A3=sqrt(s3*(s3-d3)*(s3-a)*(s3-b));
Ai(i) = 2*(A1+A2+A3)

if i < d
Ad(i) = 2*(A1+A2+A3);
phi(i) = 180*acos(dot(dir,[0;0;1]))/pi;
end

figure(1)

if (j == 1) && (m == 1)
plot_hexahedron(b_rho,b_zeta,c_rho,c_zeta,a_rho,a_zeta,SubPlots,
    j)
end

if (j == 2) && (m == 1)
plot_hexahedron(b_rho,b_zeta,c_rho,c_zeta,a_rho,a_zeta,SubPlots,
    j)
end
if (j == 3) && (m == 1)
plot_hexahedron(b_rho,b_zeta,c_rho,c_zeta,a_rho,a_zeta,SubPlots,
    j)
end

if (j == 4) && (m == 1)
plot_hexahedron(b_rho,b_zeta,c_rho,c_zeta,a_rho,a_zeta,SubPlots,
    j)
end

end
if (i == m) && (m > 1)
figure(1)
subplot(1,2,1)
histogram(Ai,1000)
set(gca,'FontSize',18)
xlabel('A_{proj}')
ylabel('N')
subplot(1,2,2)
s=scatter(phi,Ad,'.');
ylabel('A_{proj}')
xlabel('\theta_{c}')
```

```
set(gca,'FontSize',18)
xlim([0 180])
s.LineWidth = 0.1;
s.MarkerEdgeColor = 'b';
s.MarkerFaceColor = [0 0.5 0.5];
end
end
```

```
function plot_hexahedron(b_rho,b_zeta,c_rho,c_zeta,a_rho,a_zeta,
    SubPlots,j)
b2D  = [b_rho b_zeta];
c2D  = [c_rho c_zeta];
a2D  = [a_rho a_zeta];
p000 = [0 0];
p001 = p000 + c2D;
p010 = p000 + a2D;
p100 = p000 + b2D;
p101 = p100 + c2D;
p110 = p010 + b2D;
p011 = p001 + a2D;
p111 = p110 + c2D;

subplot(2,2,j)
vectarrow(p000,p100)
%legend()
axis('equal')
%xlim([-1 1])
%ylim([-1 1])

grid
hold all;
vectarrow(p000,p001)
hold all;
vectarrow(p000,p010)
hold all;
vectarrow(p100,p101)
hold all;
vectarrow(p101,p001)
hold all;
vectarrow(p001,p011)
hold all;
vectarrow(p011,p010)
hold all;
vectarrow(p010,p110)
hold all;
vectarrow(p110,p100)
hold all;
vectarrow(p110,p111)
hold all;
```

```
vectarrow(p111,p011)
hold all;
vectarrow(p111,p101)
hold all;

vectarrow(p110,p101)
hold all;
vectarrow(p110,p011)
hold all;
vectarrow(p101,p011)
hold all;

grid('off')


labels = {'000'};
plot(p000(1),p000(2),"o",'MarkerEdgeColor','black')
text(p000(1),p000(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'100'};
plot(p100(1),p100(2),"o",'MarkerEdgeColor','black')
text(p100(1),p100(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'110'};
plot(p110(1),p110(2),"o",'MarkerEdgeColor','black')
text(p110(1),p110(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'010'};
plot(p010(1),p010(2),"o",'MarkerEdgeColor','black')
text(p010(1),p010(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'011'};
plot(p011(1),p011(2),"o",'MarkerEdgeColor','black')
text(p011(1),p011(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'001'};
plot(p001(1),p001(2),"o",'MarkerEdgeColor','black')
text(p001(1),p001(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'101'};
plot(p101(1),p101(2),"o",'MarkerEdgeColor','black')
text(p101(1),p101(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;

labels = {'111'};
plot(p111(1),p111(2),"o",'MarkerEdgeColor','black')
text(p111(1),p111(2),labels,'VerticalAlignment','bottom','
    HorizontalAlignment','right')
hold all;
end
```

```

function vectarrow(p0,p1)
%Arrowline 3-D vector plot.
%   vectarrow(p0,p1) plots a line vector with arrow pointing
    from point p0
%   to point p1. The function can plot both 2D and 3D vector
    with arrow
%   depending on the dimension of the input
%
%   Example:
%       3D vector
%       p0 = [1 2 3];   % Coordinate of the first point p0
%       p1 = [4 5 6];   % Coordinate of the second point p1
%       vectarrow(p0,p1)
%
%       2D vector
%       p0 = [1 2];     % Coordinate of the first point p0
%       p1 = [4 5];     % Coordinate of the second point p1
%       vectarrow(p0,p1)
%
%   See also Vectline
%   Rentian Xiong 4-18-05
%   $Revision: 1.0
  if max(size(p0))==3
      if max(size(p1))==3
          x0 = p0(1);
          y0 = p0(2);
          z0 = p0(3);
          x1 = p1(1);
          y1 = p1(2);
          z1 = p1(3);
          plot3([x0;x1],[y0;y1],[z0;z1]);   % Draw a line
              between p0 and p1
```

```

            p = p1-p0;
            alpha = 0.1;  % Size of arrow head relative to the length of the vector
            beta = 0.1;  % Width of the base of the arrow head relative to the length

            hu = [x1-alpha*(p(1)+beta*(p(2)+eps)); x1; x1-alpha*(p(1)-beta*(p(2)+eps))];
            hv = [y1-alpha*(p(2)-beta*(p(1)+eps)); y1; y1-alpha*(p(2)+beta*(p(1)+eps))];
            hw = [z1-alpha*p(3);z1;z1-alpha*p(3)];

            hold on
            plot3(hu(:),hv(:),hw(:))  % Plot arrow head
            grid on
            xlabel('x')
            ylabel('y')
            zlabel('z')
            hold off
        else
            error('p0 and p1 must have the same dimension')
        end
    elseif max(size(p0))==2
        if max(size(p1))==2
            x0 = p0(1);
            y0 = p0(2);
            x1 = p1(1);
            y1 = p1(2);
            plot([x0;x1],[y0;y1],"black");   % Draw a line between p0 and p1

            p = p1-p0;
            alpha = 0.1;  % Size of arrow head relative to the length of the vector
            beta = 0.1;  % Width of the base of the arrow head relative to the length

            hu = [x1-alpha*(p(1)+beta*(p(2)+eps)); x1; x1-alpha*(p(1)-beta*(p(2)+eps))];
            hv = [y1-alpha*(p(2)-beta*(p(1)+eps)); y1; y1-alpha*(p(2)+beta*(p(1)+eps))];

            hold on
            %plot(hu(:),hv(:))  % Plot arrow head
            grid on
            %xlabel('x')
            %ylabel('y')
            hold off
        else
            error('p0 and p1 must have the same dimension')
        end
    else
        error('this function only accepts 2D or 3D vector')
    end
```

$wavefront.m$

```
clear all
clc
close all

P     = 1000;           %Watt
Wo    = 1e-4;   %meter
lambda  = 1064*1e-9;
discret  = 73;%247-0*220-0*230;
k  = 2*pi/lambda;
M2  =  60;

underrelax=0.3;

pfactor=1;

zo    = pi*Wo^2/lambda;
% Y  =  50000.0*Wo;
% Z  =  1.5*zo;
%Z=12*Y;

Y  =  10*Wo;
Z  =  Y;

z  =   [-Z:(Z/discret*2):Z].*ones(discret+1);
r  =  -[-Y:(Y/discret*2):Y]'.*ones(discret+1);


for m =  1:length(r(:,1))
        for  j  =  1:length(r(1,:))

               W        = Wo*sqrt(1+(z(m,j)/zo*M2)^2);
               R_ax    =  z(m,j)*(1+(zo/z(m,j))^2);

               I_gauss(m,j)=(2*P/(pi*W^2))*exp(-2*r(m,j)^2/W^2);

               U(m,j)  =  Wo/W*exp(-r(m,j)^2/W^2)*exp(-i*k*z(m,j)-i*k*r(m
                    ,j)^2/2/R_ax+i*atan(z(m,j)/zo));

```

```
            wave(m,j)=real(U(m,j));


            x_p = z(m,j);
            r_p = r(m,j);

            R = x_p*(1+(zo/x_p)^2);

            for e =1:1

                x_r = -x_p + R - sqrt(R^2-r_p^2);

                R = x_p*(1+(zo/x_p)^2);

            end

            R_plane(m,j) = sqrt(R_ax^2+r(m,j)^2);

        end
end

theshold = max(max(abs(wave)))/10;

for m = 1:length(r(:,1))
    for j = 1:length(r(1,:))


            if(abs(wave(m,j))<theshold)
                wave(m,j)=0;
            end

                    if(wave(m,j)<0)
                wave(m,j)=0;
            else
                wave(m,j)=1;
            end




    end
end

%[gradIx,gradIy]=gradient(R_plane);
%FXN=-gradIx./sqrt(gradIx.^2+gradIy.^2);
%FYN=-gradIy./sqrt(gradIx.^2+gradIy.^2);

FXN = cos(asin(r./R_ax));
FYN = -sin(asin(r./R_ax));
```

```
FXN = FXN./sqrt(FXN.^2+FYN.^2);
FYN = FYN./sqrt(FXN.^2+FYN.^2);


 for m =  1:length(r(:,1))
       for  j  =  1:length(r(1,:))
             if(z(m,j)<0)
            FYN(m,j)=-FYN(m,j);
             end
       end
end



figure(1)
subplot(3,1,1)
quiver(FXN,FYN)
c=colorbar('eastoutside');
colormap(jet)
caxis([-60  60])
subplot(3,1,2)
imagesc(R_plane)
colormap(hot)
c=colorbar('eastoutside');
caxis([0  60])
subplot(3,1,3)
imagesc(I_gauss)


figure(2)
quiver(FXN,FYN)
```

BeamShapes.m

```
clear  all
clc
close

x = ones(201).*[-2:0.02:2];
y = x';

alpha = pi()/5;
a = 0.7;
b = 1.3;
W0 = 0.5;
W0ell = 0.3;
W0_small = 0.01;
xshift = -0.7;
```

```
yshift = 0.6;

gauss = exp(-((x).^2+(y).^2).^4/W0^4);

gauss_small = 3*exp(-(((x-1.5*xshift)).^2+((y)).^2)/W0_small);

ell = exp(-((x*0.7).^2+(y*1.3).^2)/W0ell^2);

tophat = exp(-(((x).^2+(y).^2)).^8/W0^8);

ell = exp(-((x*a).^2+(y*b).^2)/W0ell);

xrot1 = zeros(length(x)).*[-2:0.02:2];
yrot1 = xrot1';
xrot2 = xrot1;
yrot2 = yrot1';

for i = 1:length(x(:,1))
    for j = 1:length(y(1,:))
        xrot1(i,j) = (x(i,j)+xshift)*cos(alpha)-(y(i,j)+yshift)*
           sin(alpha);
        yrot1(i,j) = (x(i,j)+xshift)*sin(alpha)+(y(i,j)+yshift)*
           cos(alpha);
        xrot2(i,j) = (x(i,j)+xshift)*cos(-alpha)-(y(i,j)-yshift)
           *sin(-alpha);
        yrot2(i,j) = (x(i,j)+xshift)*sin(-alpha)+(y(i,j)-yshift)
           *cos(-alpha);
    end
end


ell_rot = exp(-((x*a+xshift).^2+((y*b+yshift).^2)/W0ell));
ell_shift1 = exp(-(((x+xshift)*a).^2+((y+yshift)*b).^2)/W0ell);
ell_shift_rot1 = exp(-(((xrot1)*a).^2+((yrot1)*b).^2)/W0ell);
ell_shift_rot2 = exp(-(((xrot2)*a).^2+((yrot2)*b).^2)/W0ell);

multi = gauss_small+ell_shift_rot1+ell_shift_rot2;

figure(1)
set(gcf,'color','[1 1 1]');
subplot(2,2,1)
imagesc(gauss)
set(gca,'FontSize',20)
colormap(hot)
title('I')
axis('equal')
axis('off')
grid('off')
```

```
%xlim([0 200])
%ylim([0 200])
% subplot(2,4,1)
% %surf(gauss)
% set(gca,'FontSize',30)
% title('I')
% axis('off')
% grid('off')
% shading interp

subplot(2,2,2)
imagesc(tophat)
set(gca, 'FontSize',20)
title('II')
axis('equal')
axis('off')
grid('off')
xlim([0 200])
ylim([0 200])
% subplot(2,4,2)
% %surf(tophat)
% set(gca,'FontSize',30)
% title('II')
% axis('off')
% grid('off')
% shading interp

subplot(2,2,3)
imagesc(ell)
set(gca, 'FontSize',20)
title('III')
axis('equal')
axis('off')
grid('off')
xlim([0 200])
ylim([0 200])
% subplot(2,4,3)
% %surf(ell)
% set(gca,'FontSize',30)
% title('III')
% axis('off')
% grid('off')
% shading interp

subplot(2,2,4)
imagesc(multi)
set(gca, 'FontSize',20)
title('IV')
```

```
axis('off')
axis('equal')
grid('off')
xlim([0 200])
ylim([0 200])
% subplot(2,4,4)
% %surf(multi)
% set(gca,'FontSize',30)
% title('IV')
% axis('off')
% grid('off')
% shading interp
```

lambert.m

```
clear all
clc
clear
close all

wavelenght          = 633e-9;
frequency_medium    = physconst('LightSpeed')/wavelenght;
period_medium       = 1/frequency_medium;
envwidth_in_per     = 20;
offset_in_per       = 1.5*envwidth_in_per;
peakfield           = 1;
q                   = 4;
dt                  = period_medium/30;
dz                  = wavelenght/30;
nt                  = [1500 3500];
nz                  = 3000;
z1                  = dz*(1:(nz+1))*1e6;
z2                  = dz*(1:(nz))*1e6;
mu                  = 4*pi*10^(-7);
eps_0               = 1/(mu*physconst('LightSpeed')^2);
ref_ind             = [1.5 3.5];
materials           = size(ref_ind);
materials           = materials(2);
p=1;

for j=1:materials
    eps_r   = [ones(1,nz/2) ones(1,nz/2)*ref_ind(j)^2];
    eps_r   = [eps_r;eps_r];
    eps_r   = eps_r.';
    eps     = eps_0.*eps_r;
    plots   = size(nt);
    plots   = plots(2);
    E       = zeros(nz+1,plots);     % e-field in space
    H         = zeros(nz,plots);        % h-field in space
    S         = zeros(nz,plots);        % poynting vector

    for k = 1:plots
        for m = 1:nt(k)
            E(1,k)      = E_inc(period_medium,envwidth_in_per,...
                          offset_in_per,peakfield,q,dt,m);
            H(:,k)      = H(:,k)+diff(E(:,k))*dt/(dz*mu);
            E(2:nz,k)  = E(2:nz,k)+diff(H(:,k))./eps(1:nz-1,k)*(
                dt/dz);
        end
        S(:,k)  = -E(1:nz,k).*H(:,k);

        if      k == 1 & j == 1
            E1  = E(1:nz,k);
            S1  = S(:,k);
        elseif  k == 2 & j == 1
            E2  = E(1:nz,k);
            S2  = S(:,k);
        elseif  k == 1 & j == 2
            E3   =E(1:nz,k);
            S3  = S(:,k);
        elseif  k == 2 & j == 2
            E4  = E(1:nz,k);
            S4  = S(:,k);
        end
        p=p+1;
    end


end

subplot(3,1,1)
plot(z2,E1,z2,S1*1e3);
title('incident E- & S-field');
xlabel('position in z-direction [\mum]');
xlim([0 60]);
ylim([-2 3]);
ylabel('E & S-field [V/m],[W/m^2]');

subplot(3,1,2)
plot(z2,E2,z2,S2*1e3);
title('reflection E- & S-field n=1.5');
xlabel('position in z-direction [\mum]');
xlim([0 60]);
ylim([-2 3]);
ylabel('E & S-field [V/m],[W/m^2]');
line([nz*dz/2*1e6 nz*dz/2*1e6], [-2 3],'color',[0 0 0]);
```

```

subplot(3,1,3)
plot(z2,E4,z2,S4*1e3);
title('reflection E- & S-field n=3.5');
xlabel('position in z-direction [\mum]');
xlim([0 60]);
%ylim([-2 2]);
ylabel('E & S-field [V/m],[W/m^2]');
line([nz*dz/2*1e6 nz*dz/2*1e6], [-2 2],'color',[0 0 0]);
grid on
```

B. Operators and theorems

Cartesian Nabla operator:

$$\nabla = \vec{e}_x \frac{\partial}{\partial x} + \vec{e}_y \frac{\partial}{\partial y} + \vec{e}_z \frac{\partial}{\partial z}$$

Polar Nabla operator:

$$\nabla = \vec{e}_r \frac{\partial}{\partial r} + \frac{1}{r} \vec{e}_\theta \frac{\partial}{\partial \theta} + \vec{e}_z \frac{\partial}{\partial z}$$

Cartesian Laplace operator:

$$\Delta f(x,y,z) = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

Cartesian transverse Laplace operator:

$$\nabla_T^2 f(x,y,z) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Polar Laplace operator:

$$\Delta f(r,\theta,z) = \nabla^2 f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2} + \frac{\partial^2 f}{\partial z^2}$$

Divergence of a vector field:

$$\mathrm{div}(\vec{V}) = \nabla \cdot \vec{V}$$

Rotation of a vector field:

$$\mathrm{rot}(\vec{V}) = \nabla \times \vec{V}$$

Gradient of a scalar field:

$$\mathrm{grad}(\phi) = \nabla \phi$$

Gauss's theorem:

$$\iiint_V (\nabla \cdot \vec{F}) dV = \iint_S (\vec{F} \cdot \vec{n}) dS$$

A time-averaged function:

$$\langle \Gamma \rangle = \frac{1}{T} \int_0^T \Gamma dt$$

Laguerre polynomials:

$$\mathbb{L}_m^l(x) = x^{l+m} e^{-x} \frac{x^{-l} e^x}{m!} \frac{d^m}{dx^m}$$

Helmholtz decomposition:

$$\vec{F} = \nabla \Phi + \nabla \times \vec{A}$$

Φ...scalar potential of $\vec{F}$
$\vec{A}$...vector potential of $\vec{F}$