HORIZON 2020

HORIZON 2020 RESEARCH AND INNOVATION FRAMEWORK PROGRAMME OF THE EUROPEAN ATOMIC ENERGY COMMUNITY

Nuclear Fission and Radiation Protection 2018 (NFRP-2018-4)

Project acronym:	**SANDA**
Project full title:	**Solving Challenges in Nuclear Data for the Safety of European Nuclear facilities**
Grant Agreement no.:	**H2020 Grant Agreement number: 847552**

Workpackage N°:	***WP3***
Identification N°:	***D3.4***
Type of document:	***Deliverable***
Title:	Documentation of the design of a mass separation tool for target preparation
Dissemination Level:	***PU***
Reference:	
Status:	***VERSION 1***
Comments:	….

	Name	Partner	Date	Signature
Prepared by:	D. Studer	28	31-08-2023	
WP leader:	D. Schumann	21	31-08-2023	
IP Co-ordinator:	E. González	1	31-08-2023	Signed by GONZALEZ ROMERO ENRIQUE MIGUEL - DNI ***3708** at 05/09/2023 with a certificate issued by AC Sector Público

D3.4: Documentation of the design of a mass separation tool for target preparation

D. Studer[1], R. Dressler[2], U. Köster[3], D. Schumann[2], K. Wendt[1]

[1] *Institut für Physik, Johannes Gutenberg-Universität, 55099 Mainz, Germany*

[2] *Paul-Scherrer Institut, 5232 Villigen, Switzerland*

[3] *Institut Laue-Langevin, 38042 Grenoble, France*

February 8, 2023

With the rising demand for isotopically pure targets for the study of specific nuclear reactions, the construction of a high-throughput isotope separator is foreseen within the SANDA[1] project. Specifically the handling and purification of radioisotopes is mandatory and will be enabled by installation of the whole setup within a radioactivity monitoring area in close contact to a hot lab. In the current project phase, the design of the apparatus and establishment of a suitable facility, located at PSI, is planned. The design will be derived from experiences with the RISIKO[2] isotope separator at Mainz University, which has been successfully used for radioisotope purification and implantation, e.g. within the ECHo[3] project. It features a hot-cavity laser ion source, which is inherently element-selective and highly efficient at the same time. The laser system is based upon tunable pulsed Ti:sapphire lasers with high repetition-rate. Ions are extracted from the source region with 30 kV, followed by electrostatic beam focusing and separation with a conventional double focussing sector field magnet. After passing a separation slit, the ion beam can be re-focused to below millimeter size for implantation into detectors, collectors or targets with sub-millimeter control and resolution. The feasibility of this system is assessed by ion optics simulations. Possible improvements and the implementation for the proposed SANDA separator are discussed.

[1] Supplying Accurate Nuclear Data for energy and non-energy Applications.
[2] Resonance Ionization Spectroscopy in Collinear Geometry.
[3] Electron capture of ^{163}Ho.

Contents

1 Introduction and objective

The fundamental knowledge basis of nuclear properties and reactions finds application in a wide variety of research and industry fields, such as nuclear energy, safeguards, medicine and astrophysics. In particular, high precision data on minor actinides, e.g. ^{236,238}U, ^{245}Cm or ^{239,240,242}Pu, is in high demand within the nuclear energy community. Accordingly, also fission products in the mass regions of $A = 90 - 100$ and $A = 130 - 140$, e.g. Cs or Mo, are of high relevance for nuclear safeguards. On the other hand, promising innovative radionuclides for medical imaging techniques or cancer treatment can be found all over the nuclear chart [1]. Novel applications increasingly rely on precision data of certain nuclei, which have to be studied in dedicated experiments. A major challenge for the study of nuclear properties and reactions is the production of target isotopes in their pure form, i.e. without isotopic or isobaric contamination. While radioactive contaminant species represent the major disturbance in most cases, e.g. in half-life measurements or medical applications, even stable species may limit the achievable precision or prevent studies of nuclear reactions, e.g. neutron capture cross section measurements.

Target purification with regard to the element can often be performed by chemical methods with sufficiently high selectivity and yield [2]. Isobaric selection, on the other hand, can be achieved by electromagnetic mass separation techniques using an isotope separator. A combination can be realized by initial chemical separation followed by mass-separation within the isotope separator. An additional benefit of this method is the possibility of direct ion beam implantation into suitable target materials [3, 4] or detectors [5].

In order to make these capabilities available to the European nuclear data community, an isotope separator will be designed within the SANDA project, with the final goal to establish a new facility for target production. For high impact and support of the European nuclear data community, the "user-facility" model is envisaged, where scientists may propose experiments with individually specified target requirements. A suitable site for installation of the isotope separator is being prepared at PSI, which offers infrastructure and necessary permissions for handling highly radioactive material.

The main figures of merit of the isotope separator are selectivity, efficiency, ion throughput and versatility. Selectivity is the main reason for the additional purification step and has to fulfill the requirement specified for each individual target. It can typically be expressed as the mass resolution $R = m/\Delta m$, where the masses m and $m + \Delta m$ can just be resolved, with typical values of $R = 500 - 1000$ for single-stage magnetic mass separation. Since the starting material is often valuable and of limited quantity, the efficiency, i.e. implanted atoms/initial atoms, has to be maximized. The minimum required efficiency strongly depends on the application case and has to specified prior to implantation by assessing the feasibility of the final application for given target sizes compared to the initially available sample. The ion thoughput of the separator should be maximized in order to minimize the time required for an implantation run. In cases where high atom numbers are required, ion throughput can be a critical parameter for the feasibility of the isotope separation step. Lastly, versatility is a key requirement to fulfill the wide range of possible target specifications, with the main aspect being the universal applicability of isotope separation to a high number of chemical elements. Therefore separation approaches that are tailored to a specific element or restricted to a limited mass range have to be excluded. In many cases one has to consider a trade-off between the above-mentioned figures of merit selectivity,

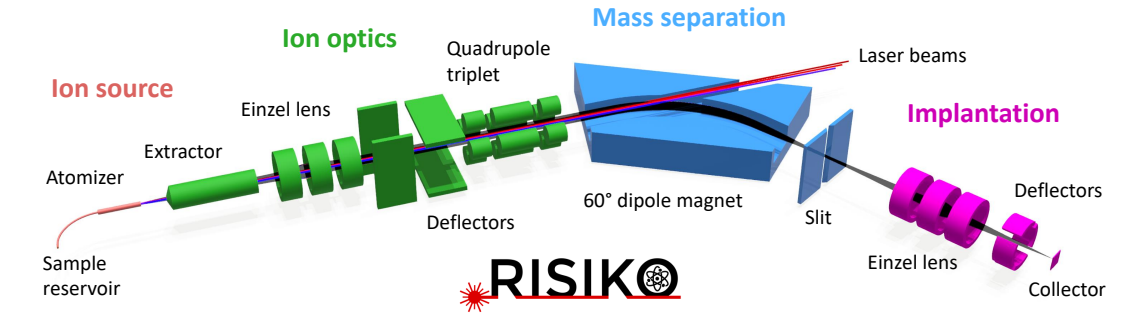
efficiency and ion throughput. Another aspect of versatility is the ability to maximize a certain figure of merit at the costs of others, offering more flexibility to fulfill use-case specific target specifications.

1.1 Ion sources

Several ion sources can be considered with regard to the above-mentioned criteria; each of them having their own benefits and drawbacks. An overview of currently used ion sources at the on-line isotope separator ISOLDE at CERN is given in [6], with a special focus on the arc-discharge ion source. Here, we give a short summary of possible options. A common feature of the sources discussed here is the generation of a thermal atomic vapor from a heated sample reservoir (or on-line target). The atomic vapor is guided to an ionization volume, where different mechanisms for efficient ionization are applied and an ion beam is extracted by high voltage electrodes.

The hot-cavity surface ion source is the simplest and most reliable option. Ionization of atoms occurs by contact with the hot walls of a resistively heated metal cavity (typically tantalum or tungsten). Although it has been shown that efficiency is considerably higher than predicted by the Saha-Equation [7], because of ion confinement in the thermal plasma within the cavity [8, 9], other options offer much higher efficiencies for most elements. Moreover, element selectivity is limited since the efficiency almost exclusively depends on each element's ionization energy. Therefore, the surface ion source is mostly used for alkaline and alkaline-earth elements, where it represents a viable option due to their low ionization energy. For more details see e.g. [8, 9, 10]. An enhancement to the hot-cavity surface ion source is given by the Resonance Ionization Laser Ion Source (RILIS). While the mechanical design of the ion source components remains essentially unchanged, it can be operated at a lower temperature and ionization occurs by high repetition-rate pulsed laser radiation. Two or more lasers are tuned to atomic resonances of the target element, which is selectively ionized in a multi-step photo-ionization process. Compared to the surface ion source, the RILIS operation provides greatly increased ionization efficiency and inherent element-selectivity. A compilation of currently available laser ionization schemes and element-specific efficiencies is given in the database hosted by the RILIS group at CERN [11]. The high number of possible laser-ionized elements underlines the versatility of RILIS operation. While solid-state lasers offer a low-maintenance option for the laser system, additional cost and operation of the lasers can be considered drawbacks of the RILIS compared to the surface ion source. Moreover, ionization efficiency starts to decline at ion currents of several 100 nA. For more details see e.g. [12, 13, 14].

A different approach is the arc-discharge ion source, which was introduced as FEBIAD and evolved into the Versatile Arc Discharge Ion Source (VADIS) [15]. It relies on electron-impact ionization of the atomic vapor within a positively biased anode cavity. Electrons emitted from a heated cathode are accelerated through a grid into the anode cavity and can additionally be confined by applying an axial magnetic field. VADIS operation is considered the most sophisticated, but also unreliable of the mentioned ion sources, due to its complexity and many degrees of freedom. However, an important advantage of the VADIS ion source is the ability to produce high ion currents at a reasonable efficiency. Recent developments aim towards a combination of arc-discharge ionization with resonant laser ionization (VADLIS)



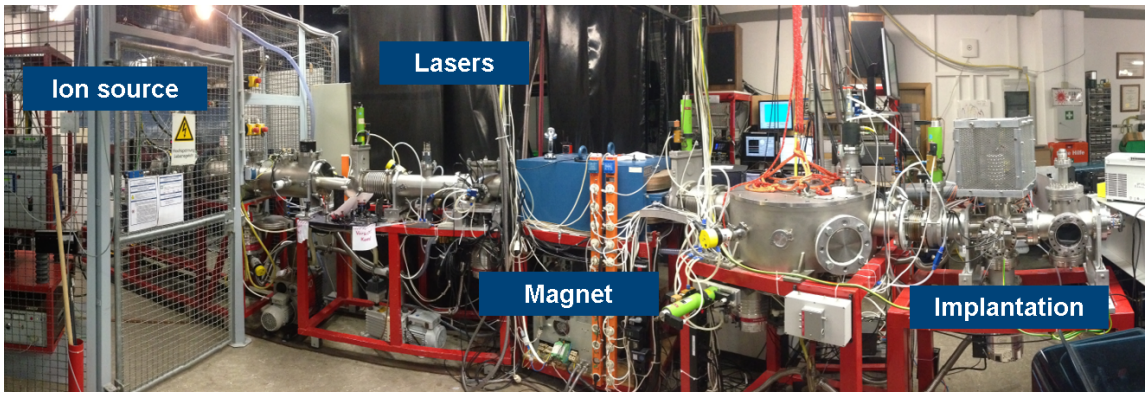


Figure 1: Top: Conceptual view of the RISIKO isotope separator. Individual sections are separated by color. Bottom: Photograph of the RISIKO Lab. Angles and distances may be distorted by the panoramic view.

[16, 17, 6].
For reasons of high efficiency, reliability and versatility, we mostly discuss the RILIS ion source here.

2 Setup overview

2.1 RISIKO

As a starting point for the SANDA isotope separator design we consider the RISIKO apparatus located at U. Mainz. It was originally designed for trace analysis of strontium isotopes following the Chernobyl disaster in 1986 [18]. It has since evolved to a multi-purpose apparatus. Typical applications include laser spectroscopy of stable and long-lived radioactive nuclei (see e.g. [19, 20]) and ion implantation in calorimetric detectors [21, 22, 5] or collector foils [4, 3].
A conceptual view of the RISIKO separator is given in Fig. 1. According to the color code in Fig. 1, the apparatus can be divided in four major stages: The ion source, extraction and transport ion optics, mass separation and ion implantation. The individual stages are briefly described in the following sections.

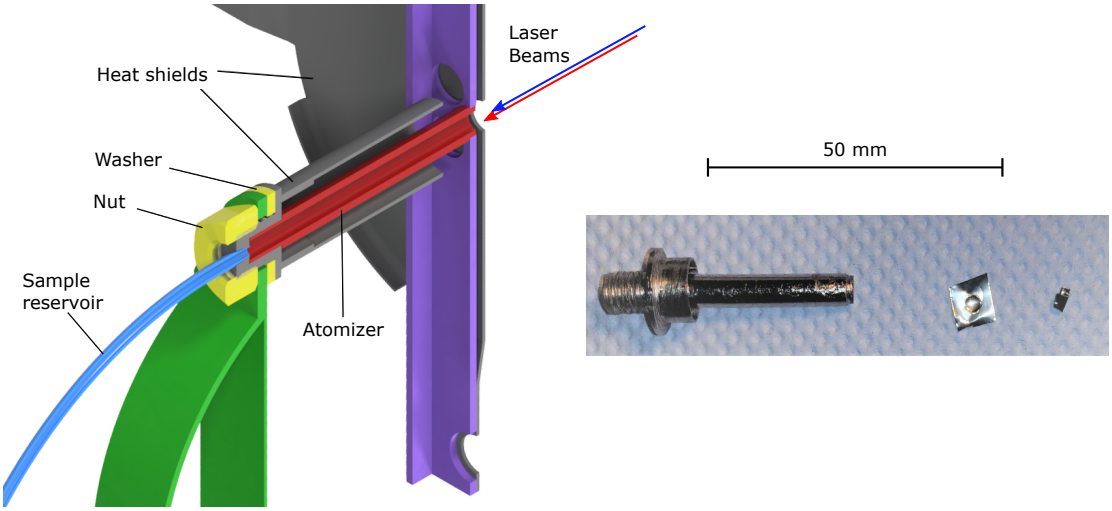


Figure 2: Left: Cross section of the hot-cavity ion source assembly of the RISIKO mass separator. The sample is placed in the sample reservoir (mass marker), which can be heated independently of the atomizer. The incident laser beams resonantly ionize sample atoms within the atomizer tube. Ions are extracted with 30 kV upon leaving the atomizer. Right: Photograph of a 5 µL sample solution droplet on a $5 \cdot 5\,\mathrm{mm}^2$ Ti foil and a folded sample "envelope", with the 35 mm long atomizer tube for scale. Figure from [23].

2.1.1 Ion source

The RISIKO separator is exclusively operated with a RILIS. The laser system is described in section 2.2. A cross-section of the ion source is shown in Fig. 2. The heart piece of the source is the atomizer (red): A tantalum tube with 35 mm length, 2.5 mm inner diameter and 1 mm wall strength. It is held in a Ta mount with a threaded back-piece and a cylindrical heat shield extending to the front of the atomizer. The atomizer assembly is mounted between a front panel (purple) and a spring (green), made of several thin Ta sheets. The front of the atomizer is pushed in the panel by a light tension of the spring. The back of the atomizer is fixed on the spring using a molybdenum washer and nut (yellow) to prevent irreversible welding of the material at higher temperatures. The spring has a slot hole allowing vertical alignment. Heating of the atomizer is achieved by applying an electric current of up to 320 A between the spring and the panel, where the latter is on local ground and the spring on a positive voltage of few volts. The panel features four holes close to the atomizer contact. The cross-section of the material bridges is similar to the cross-section of the atomizer and therefore minimizes the temperature drop towards the front of the atomizer. A heat-shield is mounted approximately 1 mm behind the panel, facing the extraction electrode.
The sample is inserted in a separate reservoir, also called *mass marker*. It is usually prepared by dropping few micro liters of a sample solution on a carrier foil (typically $3 \cdot 3\,\mathrm{mm}^2$ area). A suitable solution was found to be nitric acid, in contrast to chloride solutions which often showed poor evaporation properties in the ion source. The liquid is dried and the foil is folded to fully enclose the sample. For the formation on an atomic

vapor, the carrier foil has to act as an reduction agent in the ion source. Suitable carrier materials may vary for different sample elements. Zirconium foil with 20 µm thickness has proven to be a good choice in the lanthanide and actinide region [22].
The sample reservoir is a bent Ta capillary with 1.1 mm inner diameter and 2 mm outer diameter. The bend has a 90° angle with a 45 mm radius. The capillary is closed in the middle by crimping. On the ends, it features a conical shape which allows insertion in the back of the atomizer mount, which features a fitting conical drilling. On the other side, the sample reservoir is mounted in a massive copper ring mount, as can be seen in Fig. 3. The sample foil is inserted and pushed down the capillary to the crimping. Heating the the sample reservoir is achieved by applying an electric current of up to 120 A between the copper ring and the tantalum spring.
When the sample reservoir is heated, the sample is slowly evaporating. The atomic vapor is transported through the capillary into the pre-heated atomizer. Pulsed laser radiation is injected into the atomizer and selectively ionizes the target element. Ions are guided towards the exit by the electric field of the applied heating voltage of few volts. Near the orifice, ions are accelerated by the field leakage of the high-voltage extraction electrode and form an ion beam.
Ion source operation is usually performed according to the following procedure: (i) heating of the atomizer to a temperature that is safely above the evaporation temperature of the target element. (ii) Bake-out of the atomizer until the ion signal of common surface-ionized contamination, such as K, is dropping. (iii) Heating of the sample reservoir to a temperature slightly lower than the evaporation temperature of the target element and bake-out (iv) Slowly heating the sample reservoir until evaporation of the target element is observed. (v) optimization of lasers (spatial, temporal and spectral overlap) and ion optics. (vi) main operation phase at a stable ion current of few hundred nA. (vii) End of the separation/implantation phase by ramping the currents of the sample reservoir and the atomizer.

Further options

An extension to ion source which allows the suppression of surface-ionized species is the Laser Ion Source & Trap (LIST). The LIST structure is inserted between the atomizer front panel by moving the ion source upstream along the current feed through rods (cf. Fig. 3). It consists of two aperture entrance electrodes, a radio-frequency quadrupole and an exit aperture electrode. The idea of LIST operation is the separation of atomization and ionization volume. Therefore the two entrance electrodes act as repellers for ions and electrons emitted from the ion source. Neutral atoms entering the quadrupole structure are laser-ionized and guided to the exit by the radio-frequency field, where they are accelerated by the extraction electrode. The suppression of non-laser-ionized species in LIST operation comes at the cost of efficiency and is usually only considered when dealing with high amounts of isobaric contamination. For details on the LIST module see [24, 25, 26].
The RILIS ion source described above is fully compatible to surface ionization operation. In this case no lasers are injected. While the RILIS is operated as cold as possible, the atomizer is ramped to maximum heating current in surface ionization mode, which corresponds to a temperature of approximately 2200 °C. The sample release can still be controlled independently with the reservoir heating current, to a limit which is caused by parasitic heating of the sample reservoir through contact with the atomizer.

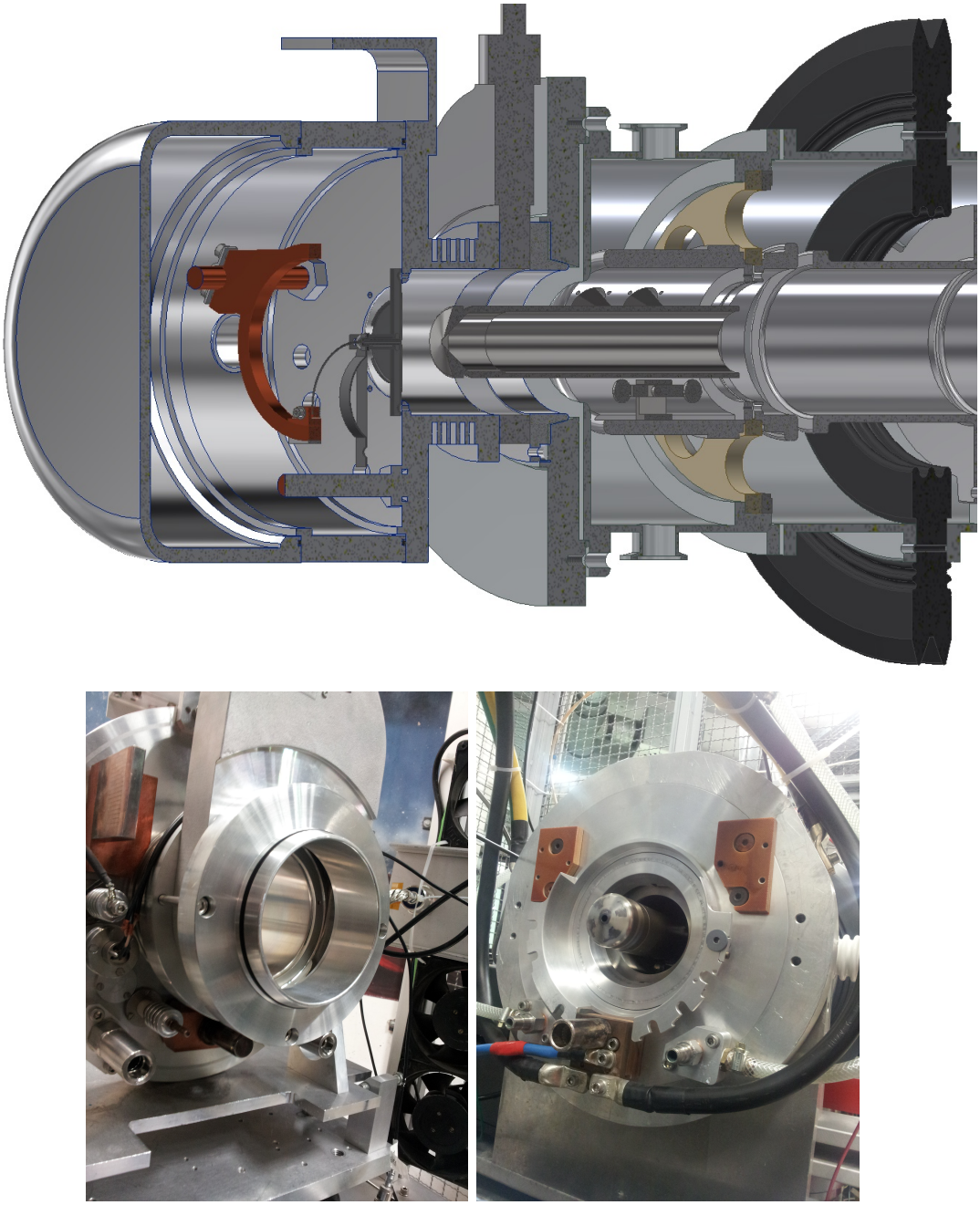


Figure 3: **Top:** Cross section of the RISIKO front-end with the standard RILIS ion source. The HV left of the insulator is 30 kV. The extraction electrode is biased with −10 kV with respect to the HV. The first electrode of the Einzel-lens (extending through the insulator) is on GND. **Bottom:** Coupling of the RISIKO ion source unit (left) to the front-end (right). High-current and water cooling are installed at the front-end and connected to the ion source unit by slide-in coupling, allowing straightforward replacement of ion source units.

2.1.2 Front-end

An important part regarding beam line design aspects is the front-end, which acts as the joint between ion source unit and ion optics. A CAD drawing of the RISIKO front-end with attached RILIS unit is given in the top of Fig. 3. It is closely adapted from ISOLDE (see e.g. [27, 28]), where the target units are exchanged by robots due to the highly radioactive environment in the ion source bunker. Therefore many parts of the infrastructure, e.g. water cooling and high current connections, are attached to the front-end rather than directly to the ion source unit. Although RISIKO, or any off-line separator, does not have to deal with the extreme conditions at on-line targets, the "quick-coupling" functionality provided by the ISOLDE front-end can be extremely useful. A photograph of the de-coupled ion source unit and the front-end is given in the bottom of Fig. 3. For the future SANDA separator we propose an ISOLDE-type front-end. The ion source unit may be exchanged or altered to apply all of the ionization techniques mentioned in Section 1.1 with reasonable modifications to the setup. An additional benefit is intercompatibility of ion source units and consequently mutual development interest with existing European facilities, such as ISOLDE, MEDICIS [1], RISIKO at U. Mainz [22], SPES at INFN [10] or the planned separators ISOL@MYHRRA at SCK-CEN and SMILES at Nantes.
As can be seen in Fig. 3, the de-coupled ion source unit allows easy access to the extraction electrode. This is important for the recovery of non-ionized sample material. The angular distribution of atoms effusing from the source is described in the general case of capillaries with arbitrary length and diameter e.g. in [29]. In the specific case of hot cavity laser ion sources this was studied in [30]. The angular distribution of the atomic beam leads to the vast majority of non-ionized atoms being deposited on the tip of the extraction electrode. This was confirmed following the manganese isotope separation project presented in [4], where radioactivity through deposition of non-ionized ^{54}Mn was measured on different ion source parts. Access to the extraction electrode and the ability to remove the tip of the electrode without causing misalignment is therefore desirable. Deposits can be removed or recovered by chemical methods and the electrode tip can be replaced when needed.

2.1.3 Ion optics

The ion optics section, reaching from ion beam extraction up the the magnet entrance, consists of the extraction electrode, an Einzel-lens, vertical and horizontal steering plates, and a quadruple triplet. Technical views of the discussed ion optics parts are presented in Fig. 4. The extraction electrode is located in the HV region of the beam line, as can be seen in Fig. 3. It is placed 40 mm downstream of the atomizer exit and is biased with -10 kV with respect to the HV (corresponding to 20 kV to GND). The tip of extraction electrode has a conical shape with an 67.5° angle to the beam axis to compensate for space-charge induced de-focusing of the beam and has a 6 mm aperture. Since most non-ionized particles form an effusing atom beam are deposited on the extractor tip [30], it can be removed to dispose or recover non-ionized sample material. The total length of the extractor is 380 mm, with the surrounding holder on the same voltage extending another 50 mm downstream. The extraction electrode is mounted on rolls within the holder, allowing translation along

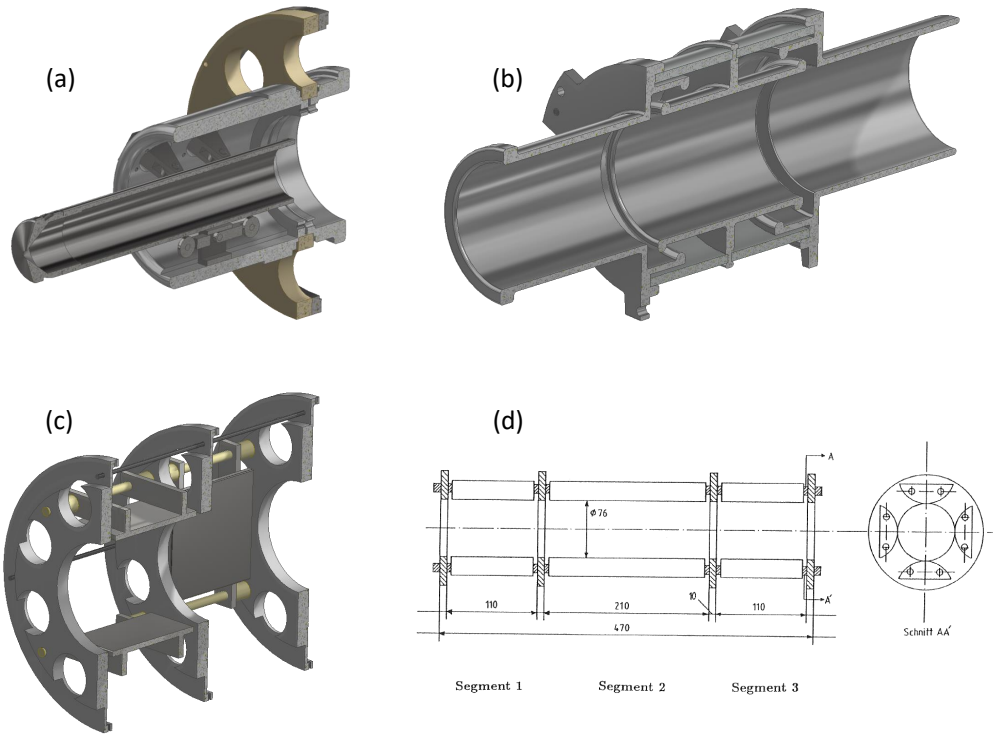


Figure 4: Technical view of the ion optics parts of RISIKO (cross section). **(a)** Extraction electrode **(b)** Einzel lens. **(c)** Deflectors. **(d)** Quadrupole triplet. In this case no 3D CAD model is available. Drawing taken from [18].

the beam axis. Moreover, the lower pair of rolls allows adjustment of the vertical tilting angle.
The first electrode of the Einzel lens follows 15 mm behind the extractor (seen on Fig. 3 extending through the HV insulator). The outer electrodes are grounded; therefore the ion beam reaches full acceleration of 30 kV upon reaching the Einzel lens. The beam energy remains unchanged by passing through the lens. The middle electrode is biased with typically 9 kV for focusing. The lens diameter of 100 mm is large compared to the ion beam diameter to minimize aberrations. After passing the Einzel lens, the ion beam should be parallel with a ≈2 cm diameter. The main purpose of the following ion optics, i.e. deflectors and quadrupole triplet, is compensation of misalignment.
The deflector consists of two pairs of flat plates with 140 mm length placed at 50 mm distance to the beam axis. The voltage applied to the plates is symmetric with opposite sign to conserve the beam energy on axis. The main source of misalignment is the atomizer, which may gain a slight angle from thermal expansion even when aligned perfectly in cold condition. Deflector voltages of <100 V are usually sufficient for compensation.
Finally, the electrostatic quadrupole triplet corrects beam astigmatism and can contribute to the formation of a parallel ion beam. A detailed description is given in [18]. The lens consists of three segments with lengths $L_1 = L_2/2 = L_3 = 100$ mm with alternately applied voltages. The quadrupole rods are half-cylinders with a diameter matching the free aperture of the lens. The double length of the middle segment compared to the outer ones leads to the focal length configuration $\pm f_1 = \mp 2 f_2 = \pm f_3$ (with opposite signs in x/y direction). In the thin-lens approximation, the focal length of the triplet in x and y direction is given by

$$f_x = \frac{f^3}{2d(f-d)} \tag{1}$$

$$f_y = \frac{f^3}{2d(f+d)} \tag{2}$$

where d is the distance between the segments' center planes. The focal length f depends on the the beam energy, the beam size (particle distance from the center) and the lens voltage. Voltages are applied in the form $U_{1,3x} = -U_{1,3y} = -(U_{2x} + U_{\text{diff}}) = (U_{2y} + U_{\text{diff}})$, where $U_{\text{diff}} \approx 0.1 U_{1,3}$ to achieve stigmatic focusing.
In RISIKO standard operation at low ion currents <100 nA, the quadrupole triplet is only used for minor corrections of ion beam astigmatism and focus optimization, which is particularly relevant for achieving small spot sized in the implantation stage. Applied voltages are in the order of 50 V.

2.1.4 Mass separation

The magnetic mass separation step is based on the circular deflection of charged particles within the field of a dipole magnet. The deflection radius is given by the equilibrium of Lorentz force and centrifugal force as

$$r = \frac{1}{B}\sqrt{\frac{2mU_0}{e}}, \tag{3}$$

with the magnetic induction B, the particle mass m and the acceleration voltage $U_0 = 30$ kV. A sketch of the mass separation principle is given in Fig. 5. A beam with finite diameter

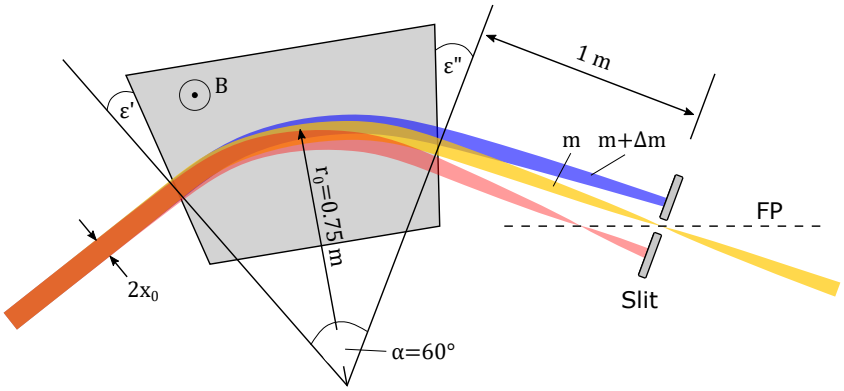


Figure 5: Sketch of the mass separation principle with a focusing dipole magnet. $\mathbf{x_0}$: beam offset; $\boldsymbol{\epsilon'/\epsilon''}$ **:** entrance/exit angle; $\mathbf{B}$: magnetic field; $\mathbf{r_0}$: nominal radius; $\boldsymbol{\alpha}$ **:** nominal deflection angle; $\mathbf{m}$: mass; $\boldsymbol{\Delta m}$ **:** mass difference; **FP:** focal plane. Figure from [23].

r_0	750 mm
α	60°
ϵ'	14.2°
ϵ''	17.2°
Pole gap	50 mm
$B_{\max}$	0.62 T
D	1440
Nominal voltage	38 V
Nominal current	92.5 A
Total weight	1035 kg

Table 1: Specification for the RISIKO magnet. Manufactured by Danfysik in 1989.

$2x_0$ is focused upon passing the magnet. With a proper choice of the magnet geometry, i.e. deflection angle α, entrance and exit angle ϵ', ϵ'' and the shape of the magnet yoke, foci in x and y direction can be influenced and matched. For the RISIKO magnet, which was purchased from Danfysik in 1989, the parameters can be found in Table 1. The focal length of the RISIKO magnet is approximately 1 m. A slit aperture with variable width is placed at the focus to separate the mass-component of choice, which is selected according to Eq. 3 by tuning the magnetic induction with the applied magnet current. The required mass resolution to separate the masses m and $m + \Delta m$ is often defined at full width at half maximum (FWHM), so that Δm can be measured at as the FWHM of a single mass peak. The mass resolution achieved at RISIKO is approximately $R \approx 700$ and scales with the beam diameter $2x_0$ and the inverse of the beam emittance ϵ according to

$$R = \frac{m}{\Delta m} = \frac{D \cdot 2x_0}{k \cdot \epsilon} \tag{4}$$

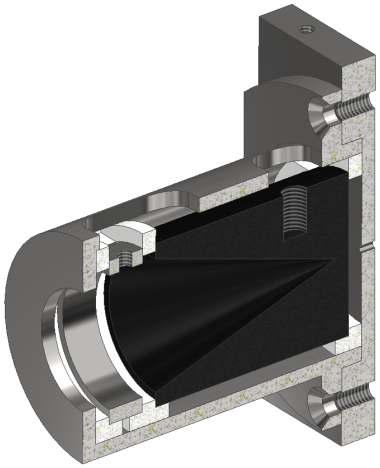


Figure 6: Technical view of the Faraday Cup placed behind the slit aperture of RISIKO.

where D is the mass dispersion coefficient and k an imaging parameter that relates the angular divergence of the beam at the magnet entrance x'_0 to the beam radius at the magnet focus x_1 with $x_1 = k \cdot x'_0$ [18]. While x_0 can be maximized to match the clear aperture of the magnet beam pipe, the emittance is mostly defined by the ion source and extraction geometry, as well as the total acceleration voltage. At RISIKO, the maximum mass for singly charged particles at 30 keV beam energy is ≈ 350 u.
The ion beam current transmitted through the slit aperture can be monitored on a retractable Faraday Cup (FC). The design was recently refined to suppress the escape of secondary particles (electrons, ions) more efficiently, allowing an accurate beam current measurement [31]. The graphite target has a low sputter yield and sputtered particles are likely to be re-absorbed in the target due to its steep conical shape. Electrons, which are more mobile and sputtered with a higher yield compared to ions, are prevented from escaping by a ring-shaped repeller electrode set to -100 V to -200 V. A technical view of the FC is given in Fig. 6.

2.1.5 Implantation

The implantation stage at RISIKO includes a post-focalization lens, a set of steering electrodes and a target Faraday Cup. A detailed description of this setup is given in [32].
With the separation slit aperture placed in the focus of the magnet, the transmitted ion beam is diverging. Sensitive focus and position control is provided by a double-stage Einzel lens and a set of electrostatic deflectors. The assembly is shown in Fig. 7. For simple alignment, the post-focalization setup is constructed as a single module made of stacked ring-shaped electrodes and insulators, where the deflectors consist of a four-segment ring with opposite voltages applied on facing electrodes. With a free aperture of 40 mm, the two-stage Einzel lens offers lower operating voltage and superior imaging quality compared to a single-stage lens. The magnification is ≈ 0.5 and the focal length $f = 230$ mm at $U_{\mathrm{foc}} = 14$ kV. At a beam energy of 30 kV, the deflection at the target plane, located 350 mm behind the princi-

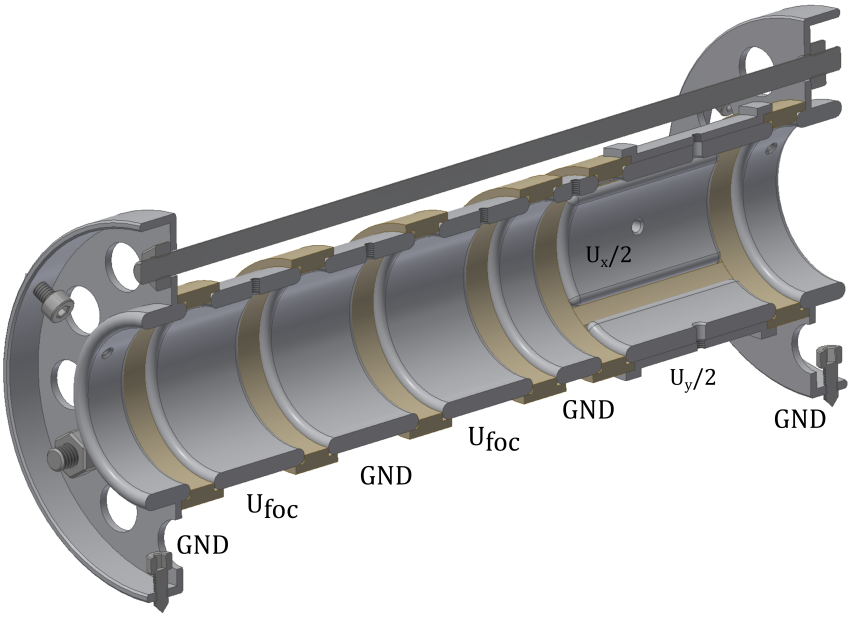


Figure 7: Technical view of post-focalization stage at RISIKO. The assembly includes a double-stage Einzel lens, a set of four steering electrodes and an exit electrode.

pal plane of the lens, is $d \approx \frac{1\,\mathrm{mm}}{125\,\mathrm{V}}$.
A technical view of the target assembly is given in [5]. The conductive implantation target is placed within a Faraday Cup. The implantation area is defined by an aperture plate, placed 1 cm in front of the target, with up to 2 cm diameter. Depending on the use case, different aperture shapes can be used. Depending on beam energy and target material different implantation depths in the orders of nanometers are reached [33, 4]. In particular in cases of shallow implantation depth, sputtering of implanted particles occurs, limiting the number of implanted atoms per area. In such cases it can be effective to work with de-focused ion beams and scanning of the beam across a large implantation area.

2.1.6 Vacuum system

An overview of the RISIKO vacuum system is shown in Fig. 2.1.6. The beam line is separated in four parts: Source (Sec. 2), magnet (Sec. 3), Switchyard (Sec. 4) and Implantation (Sec. 5). Each section can be vented and evacuated independently, allowing easy maintenance and ion source change. A 300 bar nitrogen bottle with an outlet pressure of 1.2 bar is used for venting by shutting off a section with the surrounding X.1 valves, closing the respective turbo pump with the X.2 valve, and opening the N_2 supply with X.5. When pumping down, a scroll pump (Edwards nXDS10l) is used to establish a pre-vacuum in the order of 1×10^{-2} mbar. It is connected to each section by the X.4 valves. All turbo pump exhausts are connected to a roots pump (Leybold ECUDRY 65 plus). A summary of each sections' volume, used turbo pump, and typical final pressure is given in Table 2.

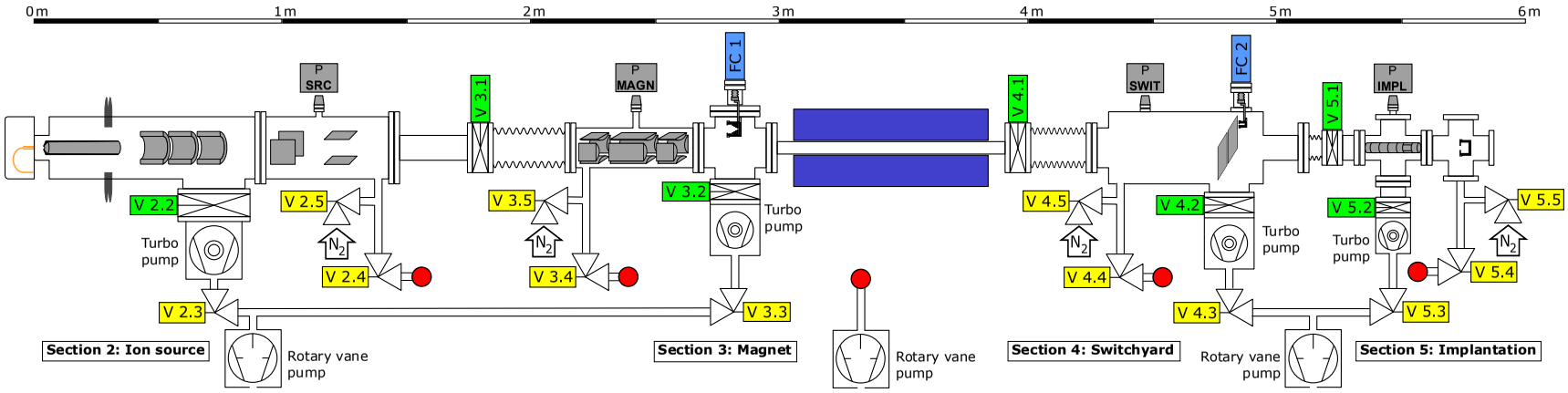


Figure 8: Vacuum scheme of RISIKO. The beam line is separated in four parts: Source (Sec. 2), magnet (Sec. 3), Switchyard (Sec. 4) and Implantation (Sec. 5). A rough length scale is given at the top. Section volumes, used turbo pumps and typical final pressure is given in Table 2.

Section	Volume (l)	Turbo pump	Final pressure (mbar)
Source	150	Pfeiffer Adixen ATP 2300 M	$1 \cdot 10^{-7}$
Magnet	80	Pfeiffer HiPace 300	$5 \cdot 10^{-7}$
Switchyard	200	Pfeiffer HiPace 300	$5 \cdot 10^{-7}$
Implantation	30	Edwards nEXT 240D	$5 \cdot 10^{-7}$

Table 2: Volume, used turbo pump, and typical final pressure for each section of the RISIKO beam line.

2.2 Laser system

The laser system used at RISIKO was specifically developed to fulfill the required specifications of laser ion sources [34]. In particular, this includes wide range tunability, pulsed operation with high repetition rate and a spectral bandwidth in the order of few GHz. Tunable lasers are a key prerequisite for resonance ionization, because the laser frequency has to matched exactly to atomic transitions of the element of interest. Ti:sapphire or dye lasers are often used for their wide spectral gain profile. While dye lasers offer higher power, they regularly require maintenance and dye change, making Ti:sapphire favorable for reliable long-term operation. Secondly, pulsed lasers are mandatory for efficient resonance ionization. Optical pumping to inaccessible long-lived atomic states has to be avoided, and therefore pulse lengths which are in the order of the atomic states' lifetime, or shorter, are required. Moreover, pulsed lasers offer high power density which allows straightforward frequency conversion, extending the accessible spectral range. The pulse repetition rate has to be high enough to expose every atom to at least one laser pulse sequence before leaving the source region by thermal movement. Finally, the spectral bandwidth has to match the Doppler distribution of thermal atoms within the source. The spectral Doppler width of an atomic transition depends on wavelength, particle mass and temperature and typically lies between 1 GHz to 10 GHz.

From case-specific requirements, different laser types have been developed over the last decade. The three commonly used Ti:sapphire laser types in Mainz are shown in Fig. 9, with specifications summarized in Table 3. The repetition rate of the lasers is defined by the pump source. The requirements for the pump laser can be specified as 15 W average power per Ti:sapphire laser to be pumped, 100 ns to 500 ns pulse length and $M^2 < 15$. Common pump sources are frequency-doubled Nd:YAG lasers with an emission wavelength of 532 nm. The Ti:sapphire laser characteristics, such as power, spectral tuning range and bandwidth vary for the different laser types. The Standard laser was optimized for the laser ion source and fulfills all requirements stated above. The more specialized designs, i.e. the grating-tuned laser and the injection-seeded laser are optimized for mode-hop-free tuning range and narrow spectral bandwidth, respectively. Major drawbacks of these designs is the lower output power for the grating-tuned laser and the requirement of a continuous-wave master laser and stabilization electronics for the injection-seeded laser. For details on the different laser types, the reader is referred to [35, 36, 37, 38, 39, 23].

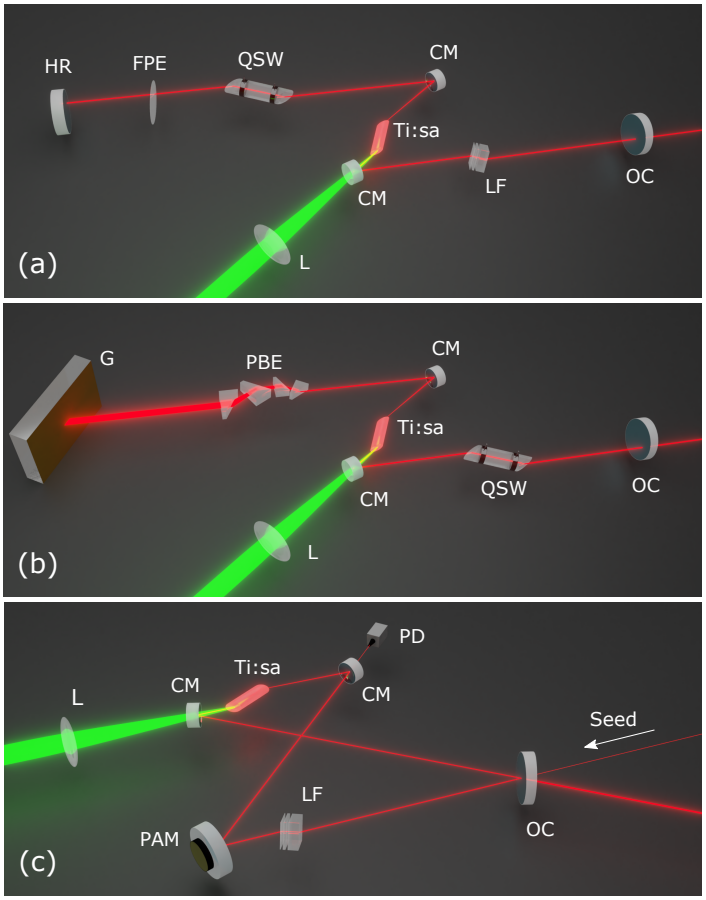


Figure 9: Layout of the different Mainz University Ti:sapphire laser types. (a) Standard laser. (b) Grating-tuned laser. (c) Injection-seeded laser. **HR:** high reflector; **FPE:** Fabry-Pérot etalon; **QSW:** Q-switch; **CM:** curved mirror; **Ti:sa:** Ti:sapphire crystal; **LF:** Lyot-filter; **OC:** output coupler; **L:** (biconvex) lens; **G:** reflective diffraction grating; **PBE:** prism beam expander: **PD:** photodiode; **PAM:** piezo-actuated mirror. Figure from [23].

Table 3: Specifications for the output of the different Ti:sapphire laser types. The values are based on the references [35, 36, 37, 38, 39, 23]. The given output powers correspond to 10 kHz repetition rate. For the injection-seeded laser, values marked with an asterisk are directly transferred from the master laser. Table adapted from [23].

	Standard	Grating-tuned	Injection-seeded
Repetition rate	7 to 15 kHz		
Pulse width	40 to 60 ns		
Average Power	3 to 5 W	1 to 2 W	3 to 5 W
Output range	700 to 1020 nm		$*$
Tuning range	100 GHz	700 to 1020 nm	$*$
Spectral bandwidth	1 to 10 GHz	1 to 3 GHz	20 MHz
Beam quality (M^2)	< 1.3		

3 Simulations

3.1 IBSimu

The capability of the RISIKO isotope separator, as described in Sec. 2.1, regarding efficiency, mass resolution and ion throughput was assessed by ion optics simulations. The open-source C++ package *Ion Beam Simulator* (IBSimu) [40, 41, 42] was chosen as a suitable tool. It incorporates space-charge density calculation, which may reveal bottlenecks at high ion current operation. IBSimu is based on a particle-in-cell approach and can be used in cylindrical, 2D or 3D geometries, including mirror symmetries. Solid bodies can be transferred to the simulation mesh by definition of boolean functions or by import of CAD files. For each solid and simulation area border, a boundary condition (Dirichlet, von-Neumann) is specified. Particles or particle beams are inserted in the form of macro-particles, carrying multiples of the microscopic particle charge, and therefore reducing computational effort. The simulation is based on the Vlasov-Poisson iteration method. Initially, the Laplace equation is solved for the given geometry. The electric field is calculated and particle trajectories are simulated. Each macro-particle deposits space-charge on the mesh, which is used to solve the Poisson equation in the following iteration. This procedure is repeated until the convergence goal is reached. In order to avoid space-charge oscillations preventing convergence, an average value for space-charge density between two following iterations is used. This so-called under-relaxation can be controlled with a weighting parameter $0 < \alpha < 1$. The number of macro-particles has to be sufficiently high for accurate space-charge deposition on the mesh. IBSimu offers several diagnostic tools for evaluation. This includes visualization of potentials, electric fields, space charge density and particle trajectories, as well as beam profile and emittance plots at user-specified planes. Geometries and particle databases can be exported and used in further simulation steps. IBSimu does **not** support magnetic field calculations. This is particularly relevant for the mass-separation step. However, external files containing the magnetic field in a (x, y, z, B_x, B_y, B_z) format, using SI units, can be imported and used for particle trajectory simulation.

3.2 Simulation geometry

The RISIKO beam line was split in several parts for the ion optics simulation in order to achieve high precision where required (ion source, slit aperture), and fast computation in large-volume sections (ion optics, magnet). A 3D simulation geometry (x, y, z) was chosen, where z is the direction of ion propagation. The (x, y)-plane lies perpendicular to the beam, with the horizontal coordinate x and the vertical coordinate y. The ion source exit lies at the origin of coordinates.
Table 4 lists the geometry sections with included electrodes (solids), mesh cell size and approximate volume. For the first two sections, up to the magnet entrance, the (x, z)-plane at $y = 0$ defines a mirror plane for the problem in order to reduce computation time. Since the magnetic field is not mirror-symmetric, this symmetry is dropped afterwards. For the slit aperture geometry, coordinates are transformed so that z remains the direction of beam propagation after passing the magnet. Components downstream of the slit aperture were not included in the simulation, since they are not relevant for the characterization of the performance of the separator in terms of efficiency and mass resolution at different ion

Section	Solids	Mesh cell size (mm)	$V(x \cdot y \cdot z)$ in mm^3
Ion source	Atomizer, Heat shield, Extrator tip	0.25	$40 \cdot 20 \cdot 115$
Ion optics	Extractor, Einzel lens, Deflectors, Quadrupoles	3	$105 \cdot 53 \cdot 2970$
Magnet	Magnet beam pipe	3	$2300 \cdot 54 \cdot 2100$
Slit aperture	Slit aperture	0.25	$40 \cdot 20 \cdot 30$*

Table 4: Simulation geometry sections with included solids, mesh cell sizes and approximate volume. The slit section is represented in transformed coordinates (indicated by an asterisk), where the z-axis points in beam direction.

currents. The simulation geometries are displayed in Fig. 10. In the simulation, the ion source was defined as ground potential, resulting in the actual ground being set to −30 kV.

3.3 Ion source & initial beam

The RISIKO ion source, as described in Sec. 2.1.1, is simulated by modeling the atomizer tube with 2.5 mm inner diameter and the front panel as a solid block on ground potential with a 2.5 mm hole (cf. Fig. 10). The initial macro-particles are generated at random positions within the atomizer volume with random velocities according to a Maxwell-Boltzmann-distribution at 2000 °C. Each macro-particle receives an additional forward energy of up to 2.5 eV in z-direction, depending on its starting position, to account for the atomizer heating voltage. The number of simulated macro-particles is in the order of 10^5, which also specifies the dynamic range of the simulation with regard to mass resolution, as will be discussed later. The charge of each particle is chosen such that a user-specified steady-state ion current is reached. A beam is extracted by the field leakage of the extraction electrode (at −10 kV) in the front region of the atomizer. Particles reaching the $z_{\max}$ plane are exported and transferred to the next simulation section.

3.3.1 Plasma model

Under vacuum conditions, most particles within the atomizer would be lost due to space-charge repulsion resulting in wall collisions. However, the high-temperature of the atomizer tube leads to the formation of a thermal plasma, which provides a confining potential and can act as an ion-guide [8, 9, 43, 44, 45]. The application of an accurate model is difficult because of the extreme conditions within the atomizer, i.e. temperature gradients, many different neutral and ionic particle species, the presence of laser radiation and consequently "laser ions", as well as thermal non-equilibrium. The above-mentioned literature can be used to assess the plasma conditions of a surface ion source under the assumptions of quasi-neutrality and thermal equilibrium at a sufficiently high temperature. Although the laser ion source is usually operated at lower temperatures, where plasma conditions do not necessarily

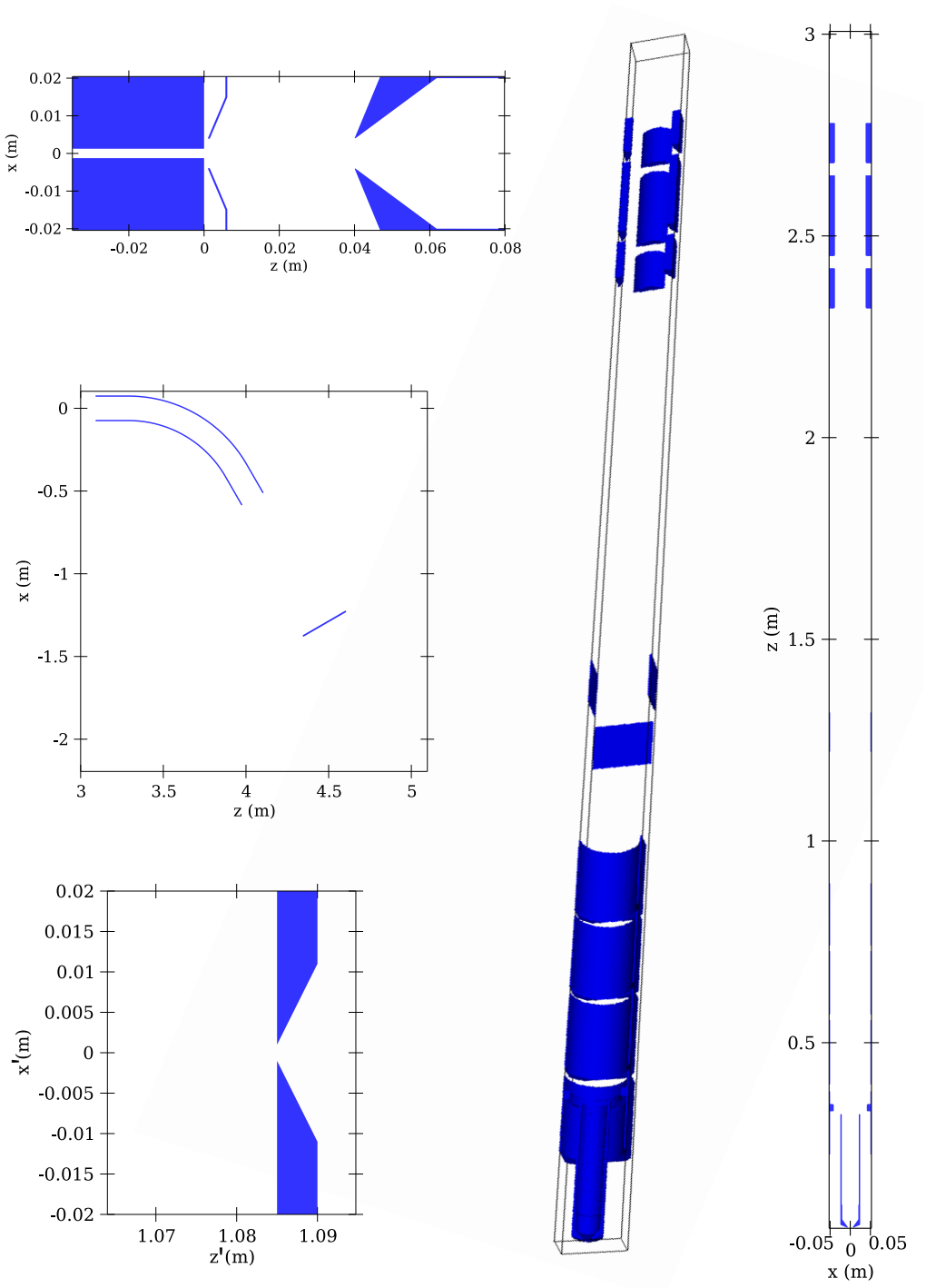


Figure 10: Simulation geometries for the different sections according to Table 4. **Upper left:** Ion source. **Right:** Ion optics. A 3D raster graphic is included for better visualization. **Middle left:** Magnet with variable position beam dump which acts as beam diagnostics plane. **Lower left:** Slit aperture in transformed coordinates. In this case $z' = 0$ corresponds to the magnet pole exit (not the beam pipe exit).

apply, we choose the surface ion source here to qualitatively represent the laser ion source. Dedicated and quantitative laser ion source plasma simulations are beyond the scope of this work.

Within the surface ion source, the density of neutral atoms n_n and ions n_i is described by the Saha-Langmuir equation. With the requirement of quasi-neutrality, the density of electrons n_e is equal to the ion density far from the walls. As a second boundary condition, the electron density at the hot surface n_{e0} is described by the thermionic emission according to Richardson's law. This problem is addressed in [45] for the case of a single hot surface and leads to a potential of

$$\frac{e\Phi(x)}{4k_BT} = \tanh^{-1}\left(\exp\left[\frac{-(x+x_0)\sqrt{2}}{\lambda_D}\right]\right) \tag{5}$$

where $\lambda_D = \sqrt{\epsilon_0 k_B T/n_p e^2}$ is the Deybe length and x_0 an integration constant. The latter can be found by satisfying the boundary condition

$$\frac{e\Phi(x=0)}{k_BT} = \ln\left(\frac{n_{e0}}{n_p}\right). \tag{6}$$

With the substitution

$$\begin{aligned} A &= 4k_BT/e \\ B &= \sqrt{2}/\lambda_D \\ C &= \ln\left(n_{e0}/n_p\right) \end{aligned} \tag{7}$$

this leads to

$$x_0 = -\frac{1}{B}\ln\left(\tanh\left[\frac{C}{4}\right]\right). \tag{8}$$

If the problem is expanded to two facing surfaces at a distance d, the potential from Eq. 5 can be extended to

$$\frac{e\Phi(x)}{4k_BT} = \tanh^{-1}\left(\exp\left[\frac{-(x+x_0)\sqrt{2}}{\lambda_D}\right]\right) + \tanh^{-1}\left(\exp\left[\frac{-(d-x+x_0)\sqrt{2}}{\lambda_D}\right]\right) \tag{9}$$

with the additional boundary condition

$$\frac{e\Phi(x=0)}{k_BT} = \frac{q\Phi(x=d)}{k_BT} = \ln\left(\frac{n_{e0}}{n_p}\right). \tag{10}$$

Solving this problem yields the integration constant

$$\begin{aligned} x_0 = \frac{1}{B}\ln\Bigg\{\frac{1}{2\left(e^C-1\right)}\Bigg[\Bigg(\Big(-2e^{\frac{C}{2}-Bd} - e^{C-Bd} - e^{-Bd} - 2e^{\frac{C}{2}} - e^C - 1\Big)^2 \\ + 4(1-e^C)(e^C-1)e^{-Bd}\Bigg)^{\frac{1}{2}} + 2e^{\frac{C}{2}-Bd} + e^{C-Bd} + e^{-Bd} + 2\frac{C}{2} + e^C + 1\Bigg]\Bigg\} \end{aligned} \tag{11}$$

with the substitution according to Eq. 7. With input parameters $n_p(\lambda_D)$ and $n_{e0}(W)$ (with the material work-function W) for a hot tantalum cavity, as given by Kirchner *et al.* [8], the potential from Eq. 9 can be plotted for different temperatures and is shown in Fig. 11. As

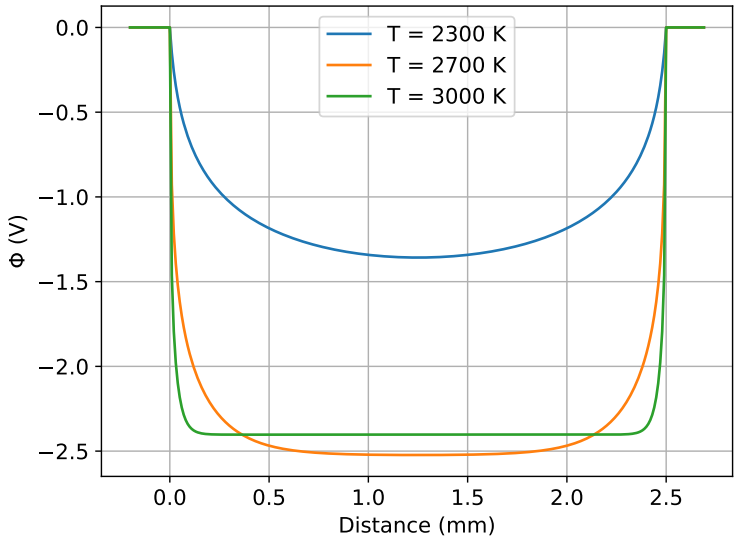


Figure 11: Plasma potential across a hot Ta cavity according to Eqs. 9 and 11, with parameters taken from [8].

stated by Kirchner, these values may not be valid for temperatures $\leq 2300\,\mathrm{K}$, because the plasma conditions are not met.

IBSimu offers a similar plasma model [41], where the temperature, plasma density and plasma potential are parameters. Both approaches suffer from some practical disadvantages: The IBSimu plasma model can only be applied in a simple way by using the library's builtin "add beam" methods, rather than adding the macro-particles in the way described in Sec. 3.3. This problem can be somewhat solved by adding the surface ions' space charge manually after each iteration. However, the results seem not to be quantitatively consistent with the "analytic" plasma model using Kirchner's parameters. Another way of tackling the problem is to fix the electric potential within the hot cavity to the results displayed in Fig. 11. This, however, leads to convergence problems in the simulation since smooth superposition of plasma potential and space-charge-induced potential is not reached. Particles are confined to a region close to the walls because of space-charge repulsion close the the atomizer axis and the steep plasma sheath approaching the wall, leading to strongly ring-shaped ion beams.

In conclusion, the IBSimu plasma model was used by adding surface ions with the same spatial distribution as "laser ions" and applying the builtin model. The expected potential well could be qualitatively reconstructed. Quantitative results about a possible ion confinement breakdown at high laser ion currents could not be obtained in this way and were not pursued further. The applied plasma mostly leads to the majority of ions being extracted rather than being lost to wall collisions. The main influence on beam extraction and focusing stems from the space-charge of the ion beam which is added on top of the plasma. At high currents, the extraction field leakage is becoming weaker, leading to less initial beam focusing, as displayed in Fig. 12. Here, we have chosen three different "laser ion" currents of ($100\,\mathrm{nA}$, $1\,\mu\mathrm{A}$, $10\,\mu\mathrm{A}$), where the former represents standard operation of RISIKO. The ring-shaped beam

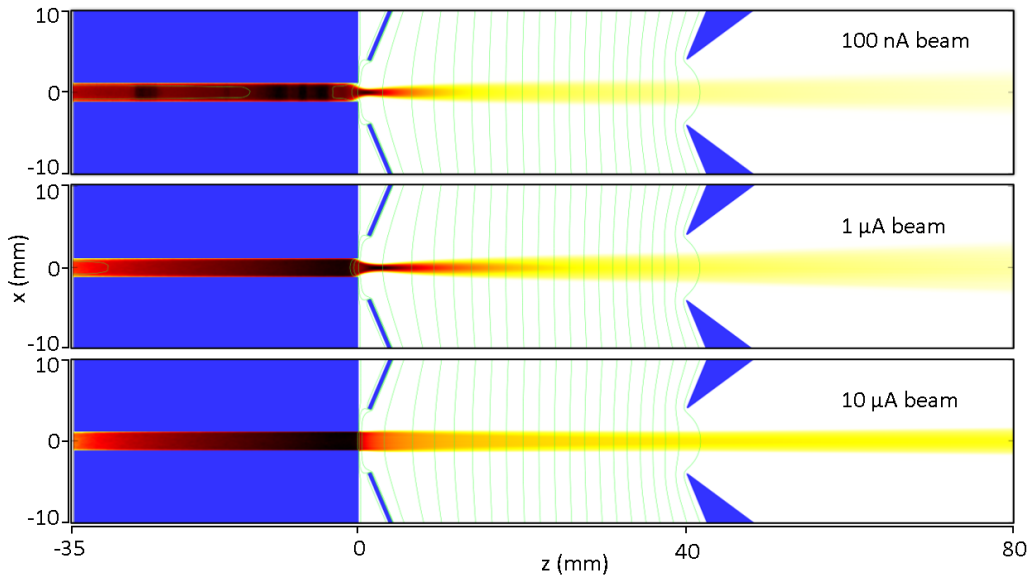


Figure 12: Extraction at different ion currents. The ion source is set to GND and the extraction electrode to −10 kV. Equipotential lines are shown in green. The ion flux is displayed qualitatively on a logarithmic color scale.

profile also occurs to some degree when using IBSimu's plasma model and becomes stronger at high currents. The influence of different parameters on this phenomenon was tested, i.e. space-charge relaxation, number of iterations, precision goal and mesh cell size. The latter has the biggest effect by far, with a finer mesh leading to a more uniform beam profile at high currents (at smaller currents the beam profile is always Gaussian). Since the mesh cell size also has to biggest impact on computation time, i.e. 9 min for a 0.4 mm mesh vs. 90 min for a 0.15 mm mesh, a reasonable value of 0.2 mm was chosen. The extracted beams are used for further assessment of the RISIKO ion optics and mass separation performance in the following steps.

3.4 Ion optics

The ion optics section spans from the interior of the extraction electrode up the magnet entrance, with the particle database imported from the ion source simulation. Free parameters of this configuration are the Einzel lens voltage U_{EL} and the quadrupole triplet voltages U_{Q13} and U_{Q2}[4]. The deflectors are not used in this case because of ideal ion source alignment, not to mention the impossibility of y-deflection due to the mirror-symmetric simulation geometry. Optimizations are initially performed towards the formation of a parallel beam with approximately 2 cm diameter, which is slightly less than the beam pipe height at the magnet entrance. It was later found that in the case of 10 µA beam current, the beam focus lies significantly further downstream compared to the two low-current cases (1.6 m from magnet exit vs. 1.09 m). As a consequence, the quadrupole triplet is used in the high-current case to form a convergent beam at the magnet entrance in order to match the focal position in

[4]We refer to the voltages of the rods in the (x, z)-plane here, with rods in the (y, z)-plane having opposite signs.

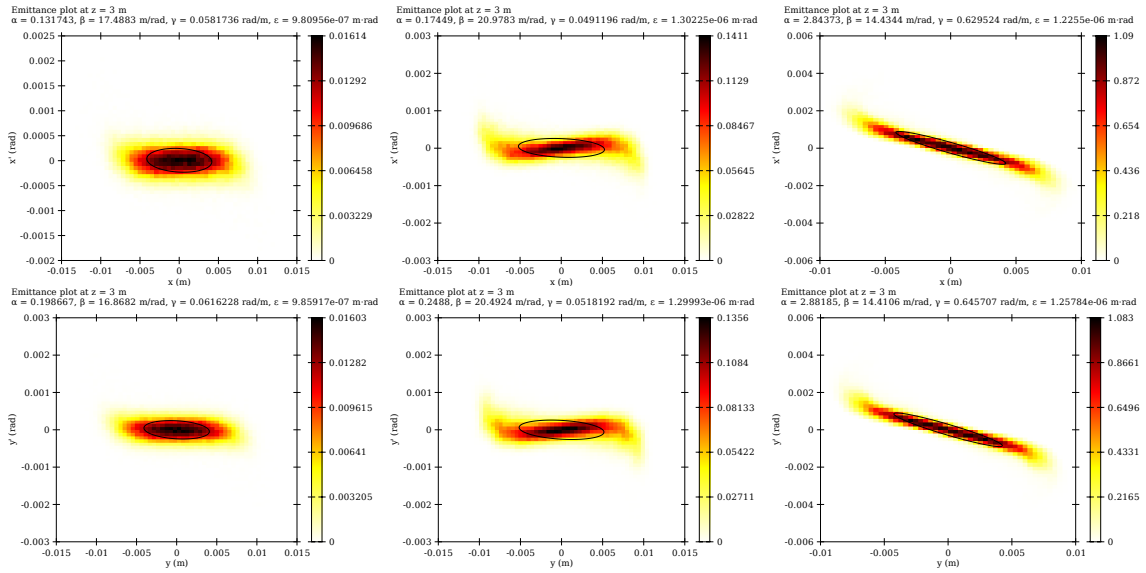


Figure 13: Emittance plots at magnet entrance in x (top) and y (bottom) directions for beam currents of 100 nA (left), 1 µA (middle) and 10 µA (right). The emittance ϵ in m rad is given on the top in each panel. The values α, β and γ are the Courant-Snyder parameters of the fitted ellipses. The color scale indicates the ion flux in A m^{-2}. The corresponding voltage settings are given in Table 5

all three scenarios. Emittance plots for the different beam current settings are given in Fig. 13. The corresponding voltages displayed in Table 5.

I (µA)	U_{EL} (kV)	U_{Q13} (V)	U_{Q2} (V)
0.1	9.8	0	0
1	10.25	0	0
10	12.5	+710	−770

Table 5: Ion optics settings for different beam currents. In all cases the ion source is set on 30 kV and the extractor on 20 kV.

3.5 Magnet

Since no magnetic field data was available for the RISIKO magnet, it was modeled according to the specifications given in Tab. 1 and the magnetic field data was afterwards simulated using SIMION 8.1. In the SIMION potential array, a grid cell size of $4 \cdot 4 \cdot 4\,\mathrm{mm}^3$ was chosen and the two magnet poles were set to arbitrary potentials of ± 1. The grid was chosen in a way that the $y = 0$ coordinate falls on a grid node. The (x, y, z, B_x, By, Bz) field data was exported and scaled in IBSimu down to the actual magnetic field values in units of T. The solid bodies representing the magnet poles were omitted in the IBSimu simulation and only the solid for the beam pipe was imported. The boundary condition for the magnetic field was set to $|B| = 0$ at all borders of the simulation geometry. The individual directions

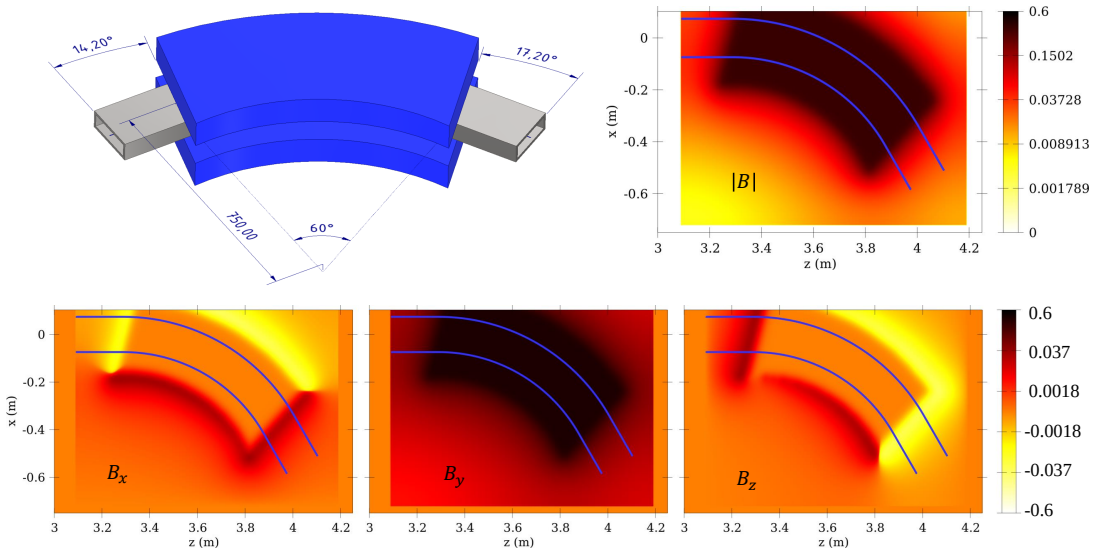


Figure 14: Magnetic field imported to the IBSimu mass separation stage geometry. **Upper left:** Simplified CAD model of the RISIKO magnet used to calculate the magnetic field. **Upper right:** Absolute value of the magnetic field in the zx-plane close to $y = 0$ (small positive offset). **Lower row:** Field components along the individual coordinate axes. The boundary condition $|B| = 0$ outside of the defined area leads to minor discontinuities, albeit at small field values. The field was scaled to transmit $m = 150\,\mathrm{u}$ and is given in units of T.

feature some negligible non-zero components and discontinuities, which do not affect the ion beam in a significant way. The magnet CAD model and the field components on the IBSimu geometry are shown in Fig. 14.

3.6 Evaluation

A critical parameter for achieving maximum mass resolution is the location of the (horizontal) beam focus, which is determined by the fitted emittance ellipse. The focal point is located where the emittance ellipse has a vertical shape, i.e. minimum spatial spread and maximum angular spread. Using the standard voltages for the 100 nA beam, we find the focus at 1.085 m downstream of the magnet pole exit. In the 1 µA case, with a parallel beam entering the magnet, the result is almost identical. However, for the 10 µA beam current the focus for a parallel beam entering the magnet was located at approximately 1.6 m. We expect that space-charge repulsion is the reason for continuous de-focusing at smaller beam diameters, i.e. in the convergent beam downstream of the magnet. To compensate this effect, a convergent beam at the magnet entrance was formed, as shown in the right panels of Fig. 13, using the quadrupole triplet.
Efficiency and selectivity evaluation can be performed in two ways: Repeated particle trajectory simulation through the magnet geometry for varying the magnetic fields B and plotting the transmitted ion current through the slit as a function of B; or evaluation of the horizon-

tal beam profile before and after passing the slit. While the former represents the practical mass scan as performed in the lab, the latter is more convenient in terms of evaluation and computation time, since it only requires a single simulation of the magnet geometry once the optimal field value is found. Moreover, it represents the actual mass-separation operation at a constant magnetic field. Evaluation is performed by analyzing all macro particles at two planes: one directly upstream and one downstream of the slit, and plotting their horizontal position and mass in histograms. The count of macro-particles per bin is converted to a steady-state ion current. The result is shown in Fig. 15. The horizontal beam profile plots show the increasing focal beam diameter with ion current. The slit opening was chosen to transmit close to 100 % of ions with mass $m = 150\,\mathrm{u}$. An interference on a level of 10^{-4} can only be observed in the highest-current case. Whether this is tolerable has to be decided for each application case. The mass resolution at FWHM achieved in the simulation is roughly $m/\Delta m = 850$ at 100 nA and $m/\Delta m = 550$ at 550 nA. Nonetheless, even in the high current case the mass resolution can still be optimized for comparably low transmission losses. Concluding this simulation evaluation, we can assess a limit steady-state ion current of 10 µA, where trade-offs between transmission and mass resolution have to be considered. It should be noted that this simulation was also performed for a parallel 10 µA beam entering the magnet and resulting optimal slit position of 1.6 m downstream of the magnet. The result roughly compares to the one presented in the right column of Fig. 15.

3.7 Limitations

The major limitation of the ion optics simulation presented in the previous section is the accurate modeling of ion source plasma conditions, as discussed is Sec. 3.3.1. While the formation of the ion beam can be qualitatively simulated, losses and space-charge limitations within the source itself are difficult to quantify. Simple plasma models cannot be applied because the laser ion source does usually not meet the thermal plasma conditions. Therefore dedicated ion source simulations remain an open task. Experimental investigations which systematically probe the ion load limit of the separator are presented in Sec. 4.

Practical accuracy and dynamic range limits of the IBSimu simulation approach arise from computation time and memory constraints. The former is dominated by the chosen geometry mesh cell size, which also has a large impact on computation time, as discussed in Sec. 3.3.1. Dynamic range, on the other hand, is influenced by the number of initial macro-particles. For example, the neighbor mass suppression can only be probed down to a factor in the order of $1/N = 5 \cdot 10^{-6}$, where N is the number of macro-particles. Nonetheless, this range can be considered sufficient because there are some practical limits on achievable mass resolution, namely *chromatic ions* and residual gas scattering. Both effects are discussed in [32]. Chromatic ions are generated within the extraction field by laser ionization, rather than inside the atomizer, and therefore experience a lower total acceleration. As a consequence of their lower energy, these ions require a lower magnetic field to pass the separator slit and therefore appear at lower masses, specifically reducing neighbor mass suppression for $\Delta m < 0$. This problem is discussed in detail in [46]. The magnitude of this effect varies and is typically in the range of 10^{-4} for $\Delta m = -1$. A possible solution is microsecond beam gating, where the deflector electrode voltage is triggered by the laser pulse and sets a transmission gate for the laser ion bunch, rejecting the earlier arriving chromatic ions.

Residual gas scattering, on the other hand, leads to beam contamination that cannot be

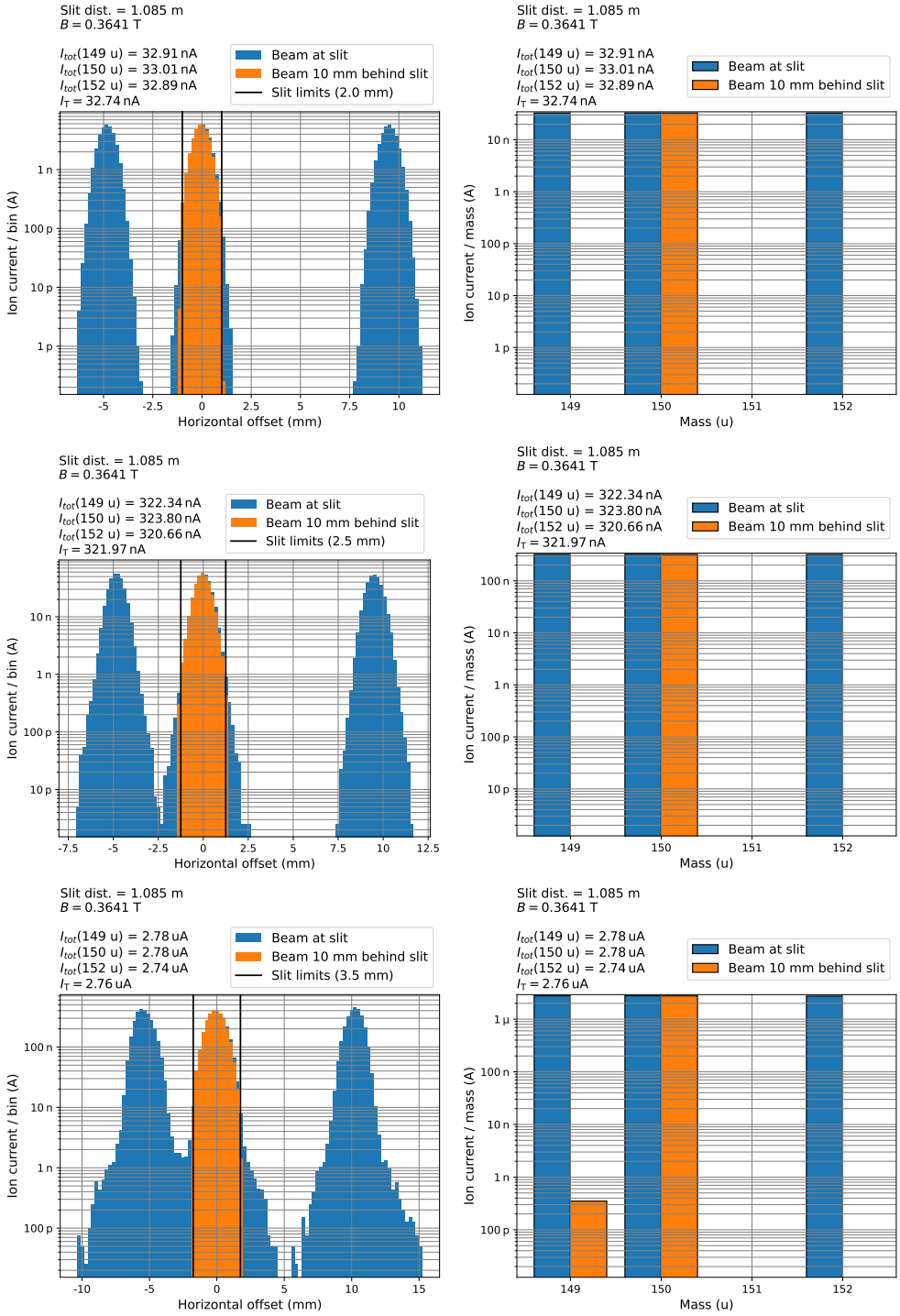


Figure 15: Evaluation of slit transmission and mass resolution for three different ion currents (rows) with an arbitrary mass composition $m = [149\,\mathrm{u}, 150\,\mathrm{u}, 152\,\mathrm{u}]$ of 1:1:1. Blue color represents the beam in front of the slit and orange color downstream of the slit. **Left:** Horizontal beam profile in bins of 0.25 mm (mesh cell size) with vertical black lines indicating the slit opening. **Right:** Mass composition in bins of 1 u.

avoided. It was assessed by Schneider [32] for the RISIKO apparatus, based on [47, 48]. At a typical pressure of 5×10^{-7} mbar it leads to a beam contamination in the order of 10^{-5} for $\Delta m = \pm 1$, which matches experimental observations.
Finally, the accurate simulation of the magnet imaging parameters was impossible due to the lack of magnetic field data or detailed geometry parameters. The magnet was modeled using a simplified geometry using the key parameters that are specified in the magnets data sheets (see Sec. 3.5). The x-focus position in the simulation was determined as 1.085 m from the magnet pole gap exit, compared to 1.043 m as given by Zimmer for the original RISIKO setup [18]. Nonetheless, we expect the magnet model to be sufficiently accurate for assessment of space-charge limitations. For the final design of the new SANDA separator, accurate field data provided by the magnet supplier should be used to determine the imaging properties and to adequately place downstream ion optic elements, e.g. post-focalization.
It should also be noted that the ion currents in the IBSimu simulation are steady-state. When using pulsed laser ion sources, peak currents are naturally several times higher than the average current. The ion bunch length is, depending on the atomizer length, typically 20 µs to 30 µs. From an ion beam time structure measurement at RISIKO, using the standard ion source and 10 kHz repetition rate lasers, the peak ion current is ≈ 6.5 times a comparable steady-state ion current.

3.8 Additional simulations

3.8.1 60 kV beam energy

As described in section 3.5, the beam emittance has to be minimized to achieve optimal mass resolution. Here, we explore the possibility of 60 keV beam energy, i.e. doubling the acceleration voltage of RISIKO, which is expected to improve beam emittance. Voltages significantly beyond 60 kV can become challenging in practice, and we therefore limit our study to the case of 60 kV here.
Similar to the setting at 30 keV, ion acceleration happens in two stages, i.e. +60 kV → +40 kV → 0 V. For ion currents > 100 nA, a slightly higher extraction voltage provides better focusing and allows the formation of a parallel beam at the magnet entrance. The voltage settings for each ion current are summarized in Table 6. Except for the increase in

I (µA)	U_{Ex} (kV)	U_{EL} (kV)	U_{Q13} (V)	U_{Q2} (V)
0.1	40	10.3	+710	−770
1	42	10.0	+410	−450
10	43	10.5	+710	−770

Table 6: Ion optics settings for different beam currents. In all cases the ion source is set on 60 kV.

extraction voltage, the settings are similar when going to higher ion currents. Emittance plots in x-direction (horizontal) are displayed in Fig. 16.

Like in the previous chapter, we observe a large angular spread in the outer area of the beam towards high ion current. However, a notable difference compared to 30 keV beam

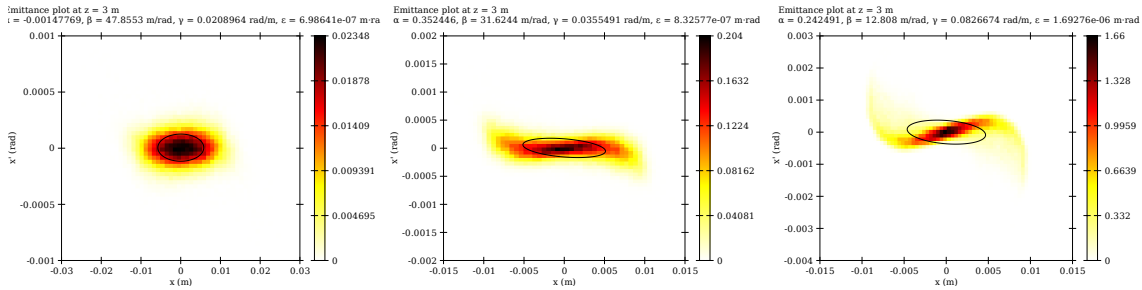


Figure 16: Emittance plots in x-direction at magnet entrance at 60 keV beam energy for beam currents of 100 nA (left), 1 µA (middle) and 10 µA (right). The emittance ϵ in m rad is given on the top in each panel. The values α, β and γ are the Courant-Snyder parameters of the fitted ellipses. The color scale indicates the ion flux in A m^{-2}. The corresponding voltage settings are given in Table 6. Emittance plots in y-direction are similar and are therefore omitted here.

energy simulation (Fig. 13) is the on average parallel beam for the 10 µA beam current. At 30 keV, the 10 µA beam needed converge at the magnet entrance in order to have the focal point in the slit plane. In the faster beam this problem does not occur, or at least to a lesser extent. Comparing the emittance at the magnet entrance, the values decreased by 30 % to 40 % for the two lower ion currents and increased by 30 % for the high current. The latter is probably due to the poor fit of the emittance ellipse to the S-shaped distribution. The better mass separation results for all ion currents are also confirmed by the visibly better mass separation shown in Fig. 17 compared to Fig. 15.

In conclusion, when applicable in practice, a higher acceleration voltage offers superior mass resolution. A downside is that higher magnetic field is required to achieve the same deflection angle (see Eq. 3). Another drawback may be higher sputtering losses during ion implantation. However, a separator capable of operating at 60 kV can always be operated a lower voltages depending on the application case.

3.8.2 Large atomizer

Another simulation was performed to assess the emittance loss when using a larger diameter ion source. A larger atomizer should in principle offer a higher ion capacity and may improve ion thoughput at the cost of emittance. Here, we chose an atomizer tube of 4.5 mm inner diameter compared to the 2.5 mm "standard" configuration, thus increasing the orifice area and atomizer volume by a factor of 3.2. Note that at this atomizer diameter the currently used laser system may not be able to provide enough power for saturation is many laser ionization schemes, however, we do not consider this aspect in the simulation. All other simulation parameters were kept similar to the 60 keV beam energy case, except that the Einzel lens and quadrupole triplet voltages had to be adjusted slightly in order to achieve a beam focus at the slit position. The ion optics settings are summarized in Table 7. Emittance plots at the magnet entrance are given in Fig. 18. Compared to the standard atomizer, emittance values increased by a factor ≈ 2.3, again disregarding the high-current case due to the poor emittance ellipse fit. The mass separation, shown in Fig. 19 is satisfactory for the beam currents of 100 nA and 1 µA. However, for the current of 10 µA, the pronounced

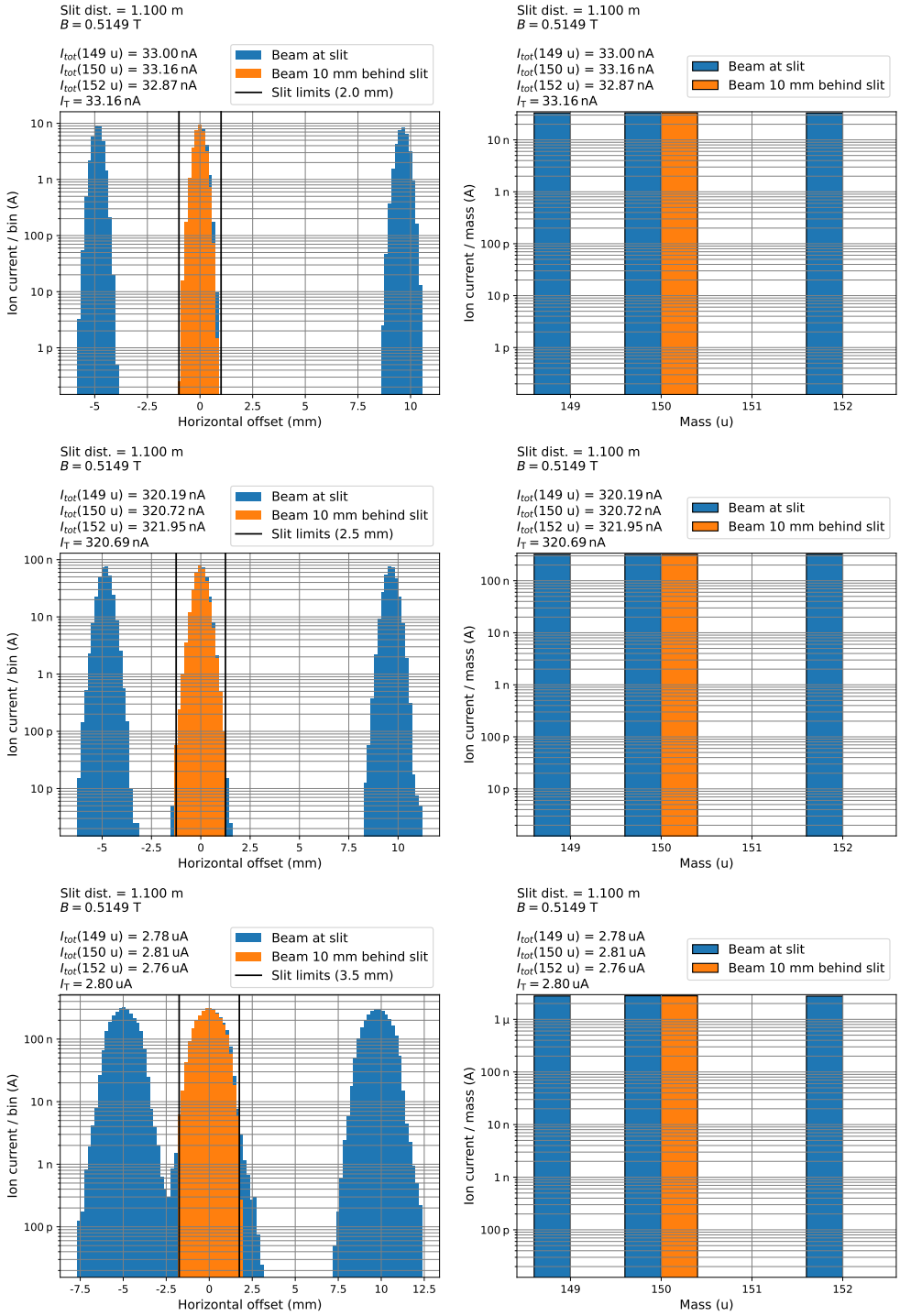


Figure 17: Evaluation of slit transmission and mass resolution at 60 keV beam energy for three different ion currents (rows) with an arbitrary mass composition $m = [149\,\mathrm{u}, 150\,\mathrm{u}, 152\,\mathrm{u}]$ of 1:1:1. Blue color represents the beam in front of the slit and orange color downstream of the slit. **Left:** Horizontal beam profile in bins of 0.25 mm (mesh cell size) with vertical black lines indicating the slit opening. **Right:** Mass composition in bins of 1 u.

I (µA)	U_{Ex} (kV)	U_{EL} (kV)	U_{Q13} (V)	U_{Q2} (V)
0.1	40	10.3	+700	−760
1	42	10.0	+360	−400
10	43	11.5	+100	−110

Table 7: Ion optics settings for different beam currents using a $d = 4.5\,\mathrm{mm}$ atomizer. In all cases the ion source is set on 60 kV.

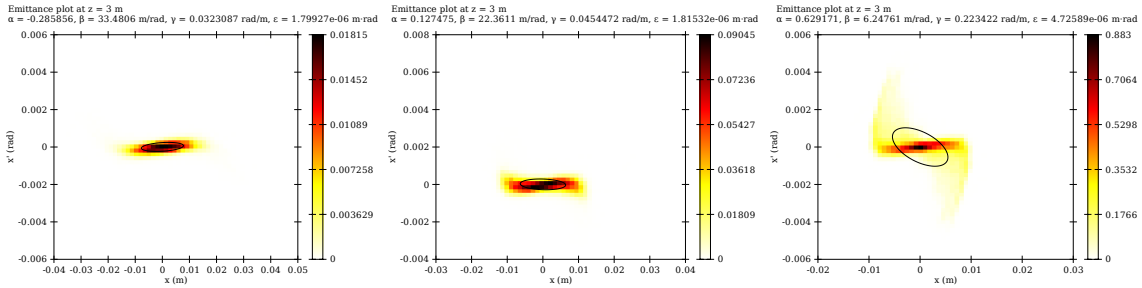


Figure 18: Emittance plots in x-direction at magnet entrance at 60 keV beam energy and using a $d = 4.5\,\mathrm{mm}$ atomizer; for beam currents of 100 nA (left), 1 µA (middle) and 10 µA (right). The emittance ϵ in m rad is given on the top in each panel. The values α, β and γ are the Courant-Snyder parameters of the fitted ellipses. The color scale indicates the ion flux in $\mathrm{A\,m^{-2}}$. The corresponding voltage settings are given in Table 7. Emittance plots in y-direction are similar and are therefore omitted here.

S-shape of the emittance distribution leads to wide shoulders in the horizontal beam cross section. The transmission is below 90 % and the neighboring mass suppression less than two orders of magnitude, which is unacceptable for most applications. In conclusion, the mass dispersion provided by the magnet is not enough to achieve clear separation at high efficiency in this case.

3.8.3 Slit ion source

Finally, a simulation of slit ion source was conducted. The vertical slit geometry of the orifice keeps the emittance in the mass separation plane at minimum, while increasing the ion source volume by extension in vertical direction. For the simulation a slot hole orifice was chosen, with a (horizontal) diameter of 2 mm and a height of 6.5 mm. The area of 16 mm is comparable to the large $r = 4.5\,\mathrm{mm}$ atomizer simulated in the previous section. In order to ensure transmission of the vertically widened beam, the extraction electrode had to be adapted. For this simulation, the extraction electrode aperture was chosen as a $3\,\mathrm{mm} \times 10\,\mathrm{mm}$ slot. The extraction electrode inner diameter was increased to 100 mm, matching the one of the Einzel lens.
The simulation of the slit ion source was limited to the most interesting case of 10 µA beam current. The ion optics were optimized for mass resolution. The voltages are given in Table 8. Note that for the quadrupole triplet, $U^x \neq U^y$ in this case. Emittance plots for the

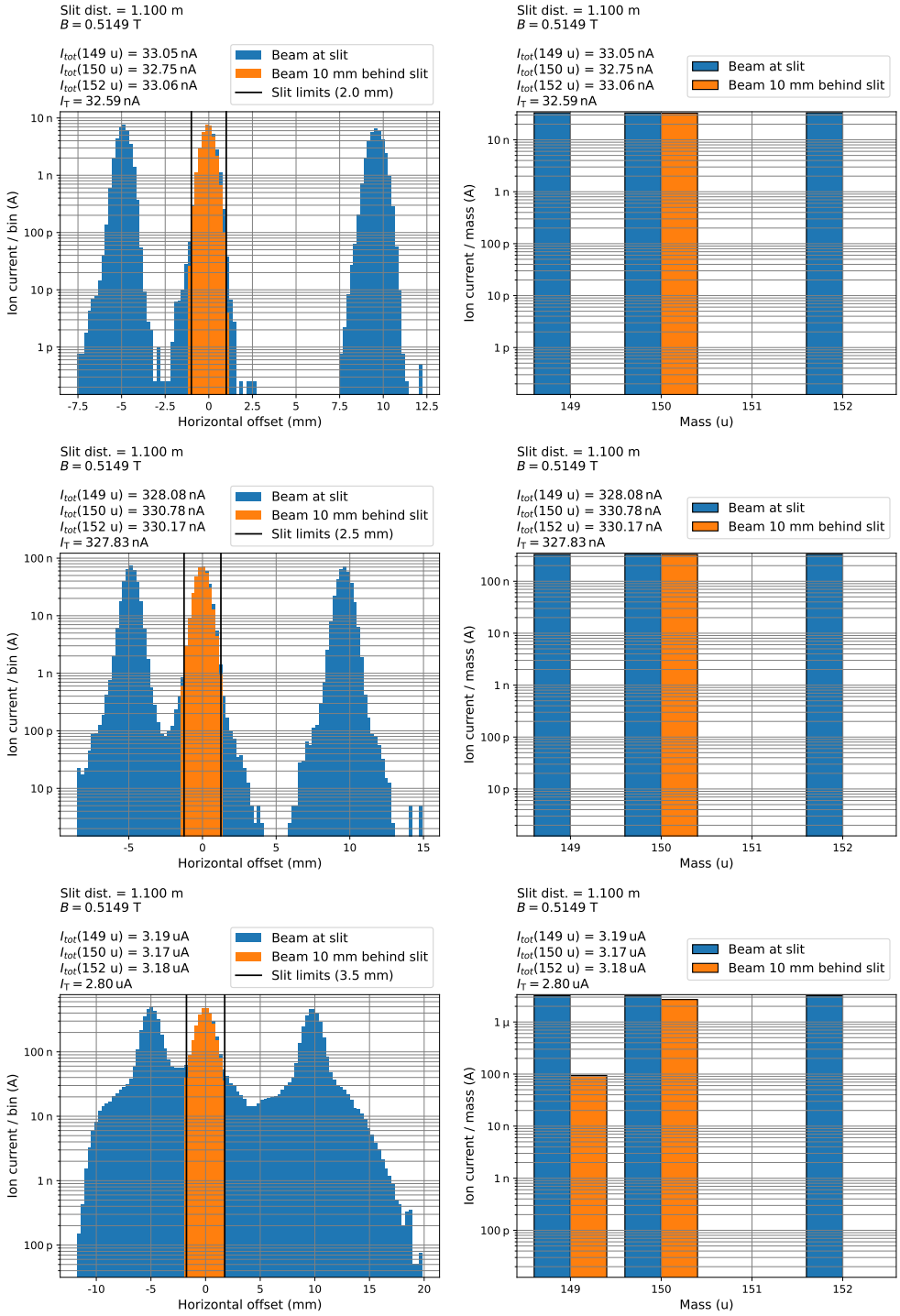


Figure 19: Evaluation of slit transmission and mass resolution at 60 keV beam energy and using a $d = 4.5$ mm atomizer for three different ion currents (rows) with an arbitrary mass composition $m = [149\,\mathrm{u}, 150\,\mathrm{u}, 152\,\mathrm{u}]$ of 1:1:1. Blue color represents the beam in front of the slit and orange color downstream of the slit. **Left:** Horizontal beam profile in bins of 0.25 mm (mesh cell size) with vertical black lines indicating the slit opening. **Right:** Mass composition in bins of 1 u.

entrance plane of the magnet are shown in Fig. 20. The beam is almost parallel in both

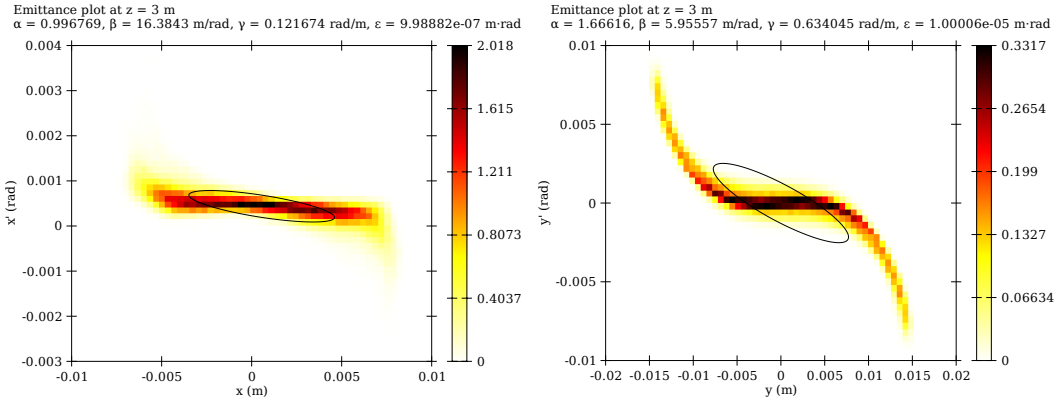


Figure 20: Emittance plots in x- (left) and y-direction (right) at magnet entrance at 60 keV beam energy using a slit atomizer and a beam current of 10 µA. The emittance ϵ in m rad is given on the top in each panel. The values α, β and γ are the Courant-Snyder parameters of the fitted ellipses. The color scale indicates the ion flux in A m^{-2}. The corresponding voltage settings are given in Table 8.

I (µA)	U_{Ex} (kV)	U_{EL} (kV)	U^x_{Q13} (V)	U^x_{Q2} (V)	U^y_{Q13} (V)	U^y_{Q2} (V)
10	40	24	+500	−1800	−1320	+550

Table 8: Ion optics settings for 10 µA beam current and 60 keV beam energy using a slit atomizer atomizer.

directions, but approximately 3 times larger in vertical direction than in horizontal direction. Using these settings the beam is focused in the slit plane 1100 mm downstream of the magnet in both directions, with a y-focus diameter of approximately 1 cm. The x-focus diameter and the resulting mass resolution can be seen in Fig. 21. Although the orifice area is the same as for the circular $d = 4.5$ mm atomizer, the mass resolution is clearly superior (cf. Fig. 19). The transmission for $m = 150$ u is close to 100 % and the neighboring mass suppression is better than four orders of magnitude. As a result, the slit ion source should be the option of choice when a sufficiently high laser power for the larger orifice can be provided. Naturally, a drawback of the slit ion source is the poor re-focusability of the transmitted beam, which renders precision ion implantation impossible. However, with the choice of a replaceable extraction electrode tip and a sufficiently large inner diameter for the extraction electrode, the slit ion source can be interchanged with the circular "standard" source, depending on the use case.

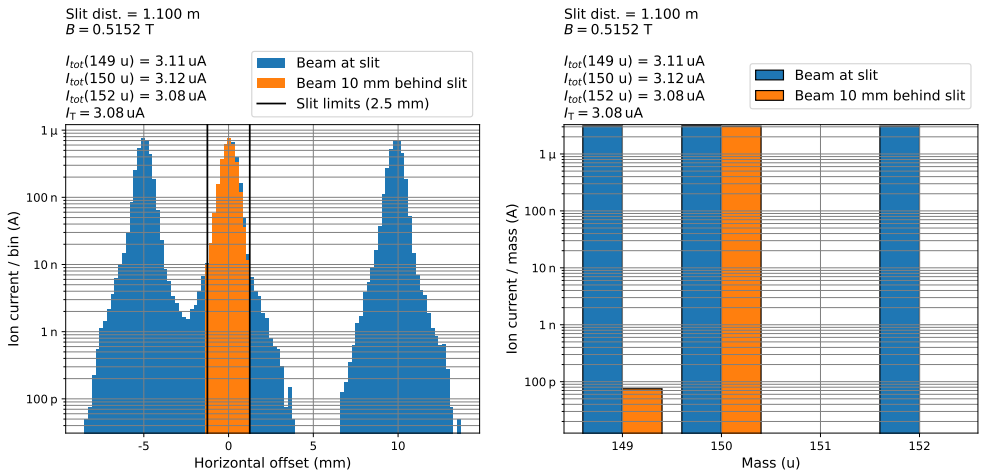


Figure 21: Evaluation of slit transmission and mass resolution at 60 keV beam energy and using a slit atomizer. The initial mass composition is $m = [149\,\mathrm{u}, 150\,\mathrm{u}, 152\,\mathrm{u}]$ with a ratio of 1:1:1. Blue color represents the beam in front of the slit and orange color downstream of the slit. **Left:** Horizontal beam profile in bins of 0.25 mm (mesh cell size) with vertical black lines indicating the slit opening. **Right:** Mass composition in bins of 1 u.

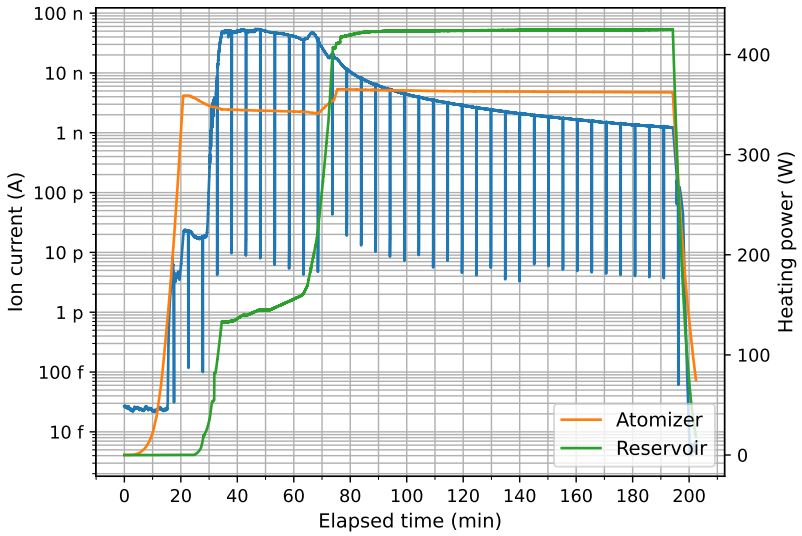


Figure 22: Efficiency measurement of a holmium sample at the RISIKO mass separator. The ion current is displayed in blue and the heating powers of the atomizer and the sample reservoir in orange and green, respectively. The sharp drops in ion current correspond to blocked ionization lasers. For details see text.

4 Experiments & Optimizations

4.1 Efficiency measurements

In addition to the simulation results discussed in the previous section, experimental methods were used to characterize the RISIKO separator and to reveal bottlenecks. In particular, efficiency loss at high ion currents was studied, which was not found to be the case in the IBSimu simulations. The total efficiency ϵ is defined as

$$\epsilon = \epsilon_{\text{Atomization}} \cdot \epsilon_{\text{Ionization}} \cdot \epsilon_{\text{Extraction}} \cdot \epsilon_{\text{Transport}} \cdot \epsilon_{\text{Detection}}. \tag{12}$$

It can be determined by measuring the ratio of detected ions N_D to the number of initial atoms N_0 in a prepared sample. For this process the sample has to be completely depleted, which can take several hours. An example efficiency measurement that was carried out in the scope of this work, using holmium as sample, is displayed in Fig. 22. Firstly, the atomizer is slowly heated to its final operation temperature, followed by heating of the sample reservoir until the release of holmiumm can be observed. Afterwards the operation parameters are optimized and the sample reservoir is heated further until the target ion current (in this case 50 nA) is reached, followed by slow heating to stabilize the ion current. Once the sample is depleted the signal quickly decays. The ion source is cleaned by increasing the heating power close to the maximum. At a comparably low ion current the measurement can be aborted. The sharp drops in current correspond to blocked lasers and reveal the contribution of laser ionization to the total ion current. Usually the laser independent ion current is subtracted for the determination of efficiency values. When using a Faraday Cup for detection, N_D is equal to the integrated ion current I over the duration of the measurement divided by the

elemental charge

$$\epsilon = \frac{N_D}{N_0} = \frac{\int I dt}{e \cdot N_0}. \tag{13}$$

Samples are prepared from an atomic absorption standard (AAS) solution, where the target element is solved in weakly concentrated nitric acid. The AAS concentration is usually specified with an uncertainty in the order of 0.1 %. Few µL of the solution are dropped on a carrier foil, followed by evaporation to dryness. This process introduces additional uncertainties in N_0. In a test case of readily prepared holmium samples in the order of 10^{15} atoms, the nominal sample size could be confirmed by NAA at the TRIGA reactor in Mainz within margins of 4 % [21]. A systematic error source in N_0 are remainders of previous samples within the ion source. For this reason efficiency measurements using blank samples are performed between actual measurements. In total, the uncertainty of ϵ is typically in the order of 10 % of the measured value, which could be confirmed by α-spectrometry measurements in the case of radium [3]. The variation in several efficiency values, on the other hand, is often higher than the uncertainty of individual measurements. This is due to varying optimization in ion optics and laser system parameters, as well as $\epsilon_{\text{Atomization}}$ from carrier foil shape and wall contact.

An experimental observation of reduced efficiency at increasing ion current was performed in the scope of manganese isotope separation [4]. Efficiencies were measured for different sample sizes, peak ion currents and measurement duration, where the latter is controlled by the ion source heating speed. Optimal efficiencies of $\epsilon = 23(7)\%$ were achieved for a slow heating phase to a maximum ion current of 10 nA. For faster heating to a similar peak current the efficiency decreased to $\epsilon = 17(2)$ %. When the peak current was increased to 500 nA, the efficiency dropped further to $\epsilon = 7(2)$ %. Both effects hint towards an ion current limitation. In the case of fast heating of the source, the contribution of surface-ionized contaminants within the beam during the most significant phase of the measurement (when the target ion current is reached) becomes higher. In the second case of increased peak ion current the laser ions themselves may limit the ion survival (vs. e.g. re-neutralization within the source) or transport efficiency.

In this work we aim to confirm these results in the well-studied case of holmium, where laser ionization efficiency was optimized over several years in the framework of the ECHo project [21, 22, 5]. Similar to the manganese experiment, peak ion currents of 50 nA and 500 nA were tested, with sample sizes of 10^{15} and 10^{16} atoms, respectively. In addition to the study of efficiency trends at different ion currents, we tested the influence of laser repetition rate. Two independent laser systems were set up, each consisting of two lasers for applying the two-step, two-color laser ionization scheme from [5]

$$0\,\text{cm}^{-1} \xrightarrow{405.5\,\text{nm}} 24\,660.80\,\text{cm}^{-1} \xrightarrow{418.3\,\text{nm}} 48\,566.95\,\text{cm}^{-1} \tag{14}$$

with the final state undergoing auto-ionization. Each laser system was operated at a repetition rate of 10 kHz and could be triggered independently, so that a repetition rate of 20 kHz could be realized at a trigger delay of 50 µs. In the case of 10 kHz repetition rate only one laser system was used. An example measurement at 10 kHz repetition rate and 50 nA peak current is shown in Fig. 22. An overview of all achieved efficiencies for different operation parameters is shown in Fig. 23. The uncertainty on each measurement was estimated as 7 % of the efficiency value from adding the uncertainties of 4 % in N_0 [21] (cf. Eq. 13) and 5 %

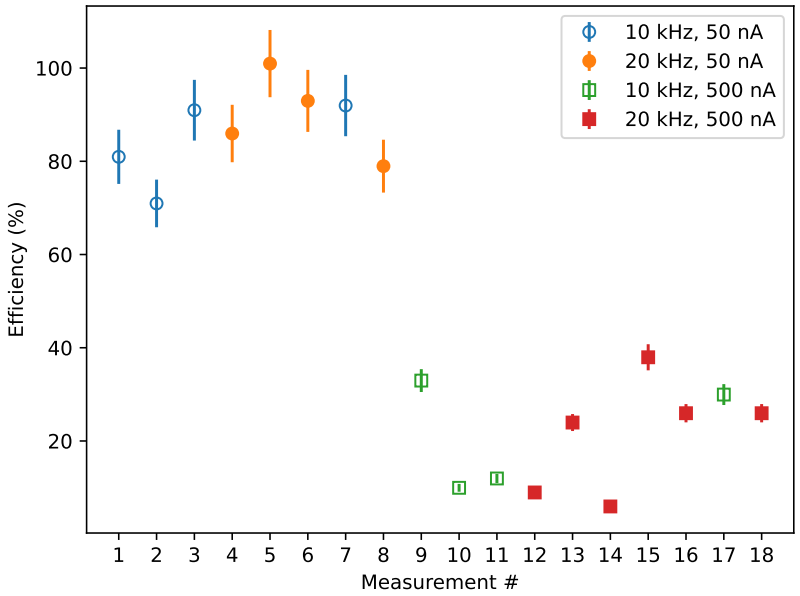


Figure 23: Overview of the holmium efficiency measurements at different ion currents and laser repetition rates. The results are sorted by measurement date. Values are given in Table 9.

from the ion current measurement [5] in quadrature. A detailed overview is given in Tab. 9. The trend observed in the manganese measurements [4] is clearly reproduced. Efficiency values drop by about a factor of four when increasing the ion current from 50 nA to 500 nA. The influence of the laser repetition rate is not significant, which is a bit puzzling considering the an increase in ion current between 70 % and 90 % when unblocking the second laser system. Since the 10 kHz and 20 kHz repetition rate measurements were performed with similar maximum ion currents, we assume the ion pulse peak current is half for the 20 kHz measurements compared to those performed at 10 kHz. The similar decrease in efficiency when going to 500 nA strongly hints towards a non-linear decrease in of efficiency with increasing ion current. Consequently more measurements are required in order to characterize the ion source behavior and finding the optimal ion current for highest throughput with good efficiency. However, this may strongly vary by sample composition and target element, since surface ionization of contaminants within the source can strongly affect the total current. In the case of holmium discussed here we can conclude that an ion current of 50 nA is safely below any limitation, since the efficiencies of $> 80\,\%$ are among the highest measured at RISIKO so far.

4.2 Laser ionization efficiency

One of the most important aspects of total efficiency is the (laser) ionization efficiency. An independent determination $\epsilon_{\text{Ionization}}$ allows to further de-convolute the individual factors in Eq. 13 and targeted optimization. For measurement we take advantage of the two independent laser systems, as described in the previous section. The two laser pulse sequences, each applying the scheme from Eq. 14, can be triggered with a variable delay with respect

Sample atoms	Rep. rate (kHz)	Meas. #	I_{max} (nA)	Duration (h)	ϵ %
$1.1 \cdot 10^{15}$	10	1	53	3	81
		2	55	3	71
		3	50	4.5	91
		7	50	3	92
	20	6	55	3	86
		5	55	3	101
		4	75	3	93
		8	50	3	79
$1.1 \cdot 10^{16}$	10	9	520	2	33
		10	400	3	10
		11	200	4	12
		17	300	3	30
	20	12	100	3	9
		13	500	4	24
		14	100	4	6
		15	200	4	38
		16	300	3.5	26
		18	360	3	26

Table 9: Overview of holmium efficiency measurements Sorted by sample size and laser repetition rate. The chronological order of measurements is indicated by the measurement number. A graphical overview is given in Fig. 23.

to each other. This allows to perform a laser ionization efficiency measurement according to [49]. After the first laser pulse sequence, the second pulse sequence is used to probe the amount of remaining atoms within the source and therefore measure the laser ionization efficiency. The pulse delay has to be sufficiently long for the population in the short-lived intermediate state of the ionization scheme to decay back to the ground state, "resetting" the atoms; and short enough for atoms to remain within the source (and not leave by thermal movement). Here, we chose 200 ns delay to fulfill both requirements. A total of 13 "instant efficiency measurements" were performed at varying laser ion currents between 100 nA and 1 µA. An ion current dependence is not significant and the measurements average to a value of 76(7) %. This value is consistent with the total efficiency of $69(5)_{stat}(4)_{sys}$ reported in [5], where the same ionization scheme was used. However, it seems to be somewhat lower than the total efficiency measured at low currents presented in the previous section, which averages to 85(9) %. This discrepancy hints towards a systematic error in sample preparation, where the sample size is slightly underestimated. Since the efficiency measurements from

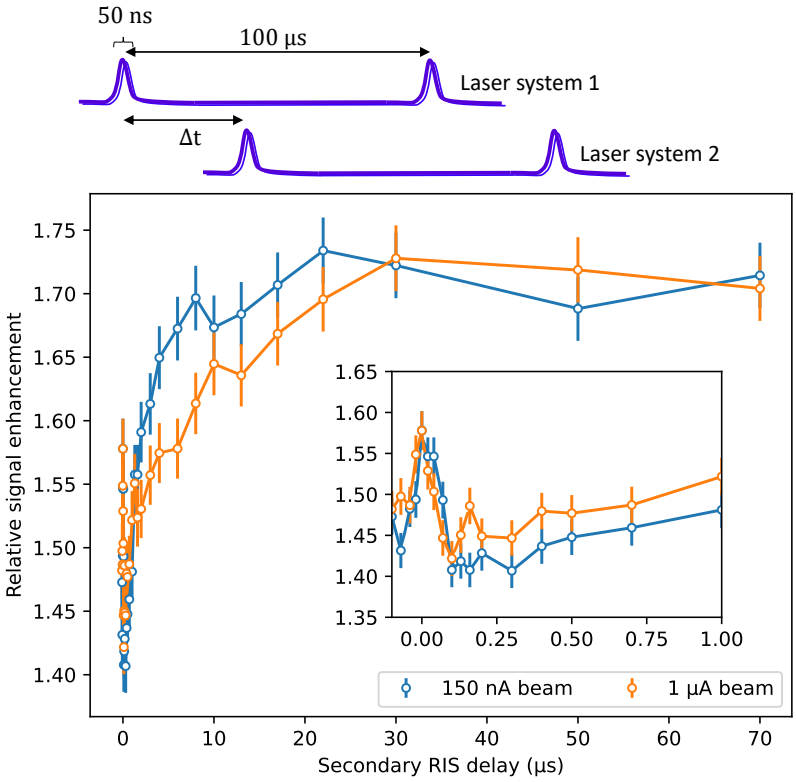


Figure 24: Atomizer regeneration time probed by delayed secondary ionization laser system.

Fig. 23 and Table 9 show a clear dependence on ion current, whereas the laser ionization efficiency measurements are constant, the drop in efficiency can be attributed to $\epsilon_{\text{Atomization}}$, $\epsilon_{\text{Extraction}}$ or $\epsilon_{\text{Transport}}$, where the latter is unlikely to have a large impact according to the ion optics simulations performed in Sec. 3. The focus for optimization of ion throughput therefore lies at atomization and ion survival within the source.

4.3 Time structure measurements

Apart from measuring laser ionization efficiency, as discussed in the previous section, a variable delay between two laser ionization sequences can be used to probe the signal enhancement of a secondary pulse sequence for a given delay. The temporal evolution of laser enhancement with respect to an initial laser ionization pulse offers another point of view on the effectiveness of 20 kHz laser repetition rate operation, since this measurement probes the "refill time" of the atomizer after applying a laser pulse. Fig. 24 shows the relative signal enhancement of a secondary laser system as a function of its temporal delay with respect to the primary laser system. To account for signal decay with time, the ion current at zero delay was measured after each data point and used to linearly correct the measurement. The two laser systems were operated at similar output power and optimized individually for optimal ion signal. For the high-current measurement, laser system 1 yielded an ion current of 660 nA and laser system 2 a current of 590 nA, with the 50 µs delay setting yielding a current of 985 nA. When both lasers were blocked the current was 2 nA. The signal enhancement is normalized to the average ion current measured when blocking one laser system.

The signal increase in Fig. 24 at zero delay is caused by the coinciding laser pulse sequences and as a result higher laser power. At a delay of approximately 200 ns the signal minimum is reached due to full temporal separation of the laser pulse sequences and relaxation of any excited atoms from the first pulse sequence. At this delay the "instant ionization efficiency", as described in the previous section, was measured. The signal enhancement then increases towards longer delays, with a maximum factor of 1.7 after roughly 20 µs. At 10 kHz repetition rate, a coincidence of laser pulse sequences would be reached again at a delay of 100 µs, whereas 50 µs delay effectively mimics 20 kHz repetition rate operation. The fast saturation of signal enhancement, both at an ion beam current of 150 nA and 1 µA, suggests a full "regeneration" of the atomizer and therefore maximum feasibility of 20 kHz repetition rate operation. On the other hand, a signal enhancement factor of 2 would be expected at full regeneration, rendering the result somewhat less conclusive. Moreover, the efficiency measurements presented in Sec. 4.1 show no significant benefit from 20 kHz operation compared to 10 kHz. This inconsistency of efficiency results and time structure measurements has to be studied in more detail in the future. For now we conclude that 20 kHz repetition rate operation looks promising, since the ion beam load can be distributed more equally in time, reducing peak currents. One explanation for an apparent discrepancy would be a rapid drop of efficiency when surpassing a certain ion current. When the respective steady-state current to the pulsed currents of 500 nA in 10 kHz and 20 kHz both lie above this limit, our experimental observation can be explained. However, due to the low reproducibility of efficiency measurements, conclusions can only be drawn with limited confidence. Obtaining good statistics in efficiency vs ion current measurements is a high-effort task, since several measurements, each taking several hours, have to be performed for each set of operation parameters.

5 Conclusion

From the simulations presented in section 3, we conclude that a RISIKO-like separator is capable of handling steady-state beam currents in the order of 10 µA. The peak current of a 10 kHz pulsed ion beam extracted from a typical laser ion source is equal to roughly 6.5 times a comparable steady-state current, which was determined in an ion beam time structure measurement using an atomizer with 35 mm length and 2.5 mm inner diameter. This means our 10 µA beam current simulation should represent pulsed currents of at least 1.5 µA. However, the simulation results stand in contradiction to the efficiency measurements presented in section 4.1, where a decline in efficiency was already observed at a pulsed current between 50 nA to 500 nA. A similar observation was made in [4]. The discrepancy between experiment and simulation most likely originates from the ion source itself. It is not possible to accurately model a laser ion source using IBSimu, for the reasons discussed in section 3.7. The overestimation of ion throughput or ion survival within the source is probably due to a limited ion capacity of the hot cavity. This problem can be tackled by up-scaling the ion source volume, e.g. by using a cylindrical source with a larger inner diameter. On the other hand, a larger orifice increases beam emittance and therefore impedes on the mass resolution. The extent of this trade-off in resolution was studied in section 3.8. Firstly, the acceleration voltage was increased to 60 kV, which is practically feasible and reduces beam emittance by ≈ 30 %. A complete neighboring mass suppression, down to the dynamic range of the simulation, could be achieved even for the 10 µA beam current case. In a next step, the atomizer inner diameter was increased from 2.5 mm to 4.5 mm, thus increasing the orifice area by a factor of 3.2. The feasibility of such a large orifice for a laser ion source is questionable since the laser power that is required for saturation of all transitions in an ionization scheme has to be scaled accordingly. Compared to the standard ion source, the simulation for the two low-current cases still showed a complete neighboring mass suppression at close to 100 % transmission through the slit aperture, however, the high-current simulation showed poor results for mass resolution. A slit ion source can be used to mitigate this problem to some extent. The horizontal emittance (and thus mass resolution) is kept at a minimum while the orifice area is increased in vertical direction. The simulation in 3.8 showed superior mass resolution results compared to the cylindrical source with the same orifice area. An extreme case of such a separator design is presented in [50]. Major drawbacks are the large magnet beam pipe requirement due to divergence in vertical direction and a drastically limited ability for re-focusing and controlled ion implantation with high spatial resolution.

Another approach for increased ion throughput is lowering the peak ion current by operating the laser system at a higher repetition rate. This was studied experimentally for 20 kHz repetition rate and is presented in section 4. While no obvious increase in overall efficiency could be observed (see section 4.1), the studies in section 4.3 point towards feasibility of up to 40 kHz, since the "regeneration" time of the atomizer is on a timescale of ≈ 25 µs. Of course, operating the lasers at a higher repetition rate while maintaining a similar pulse energy for saturation requires a higher output power of the lasers. A high-power laser prototype was developed for this purpose. Thermal lensing effects of the standard Ti:sapphire laser system, as described in section 2.2, were characterized and the results were considered in a new design featuring two gain crystals, which is presented in the appendix 6.1. The new laser can provide up to two times the output power of the standard Ti:sapphire laser with similar beam quality.

In conclusion a design close to the RISIKO separator is an appropriate choice for a new separator when efficiency is the main focus. The RILIS ion source is unsurpassed in efficiency but suffers from a practical limit in ion throughput. Larger orifice areas decrease the ion beam quality and possibly laser ionization efficiency, since the required laser power scales with orifice area.
Technical drawings of essential parts of the RISIKO separator and newly designed parts are included in the appendix 6.2. Possible upgrades include larger ion optics for reduced imaging errors for large ion beams, and the application of 60 kV acceleration voltage. The latter improves beam emittance and therefore mass resolution even at a high ion current. However, it also implies a higher magnetic field for deflection of the fast beam, e.g. 0.8 T at 750 mm deflection radius and a particle mass of 290 u (cf. Eq. 3). Additionally, use of a slit ion source should be enabled by using a larger diameter extraction electrode with replaceable tip. Finally, mass dispersion can be improved by choosing a greater deflection angle, e.g. 90°. In order to incorporate a different separator magnet in the design, the simulation code is included in the appendix 6.3. Instructions on how the code can be altered to adjust the design are given preceding each code listing. Technical drawings and simulation files are accessible at https://seafile.rlp.net/d/cff49f16f96a4777950d/.

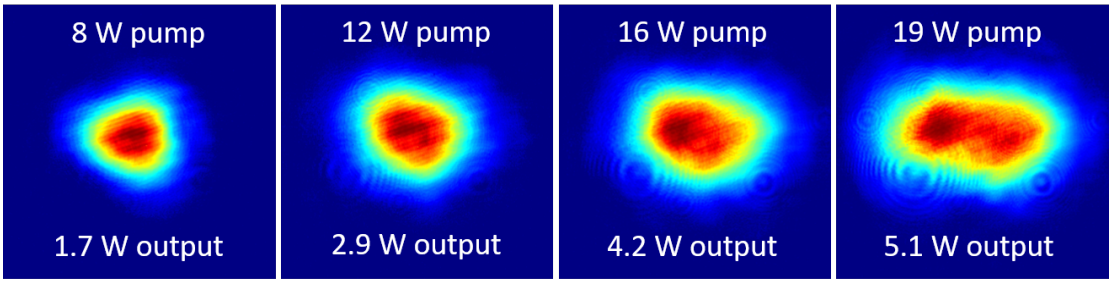


Figure 25: Standard Ti:sapphire laser beam profiles at different pump power settings. The higher transverse mode structure at high pump power is clearly visible.

6 Appendix

6.1 High-power laser system

The laser system that is currently in use at RISIKO is presented in Sec. 2.2. In order to increase the repetition rate of the laser system from 10 kHz to 20 kHz while maintaining a similar pulse energy, the output power has to be doubled. Since the general resonator layout of the standard Mainz Ti:sapphire is already optimized for high power output, the pump power has to be increased, which introduces new difficulties. Although the conversion efficiency shows a linear trend with increasing pump power, the output beam eventually exhibits additional transversal modes, visible in elliptical shapes of the output beam profiles of a standard Ti:sapphire laser at four different pump power settings shown in Fig. 25. Higher transverse modes strongly influence the focusing capabilities of the output beam and can be quantitatively described with the beam quality factor M^2. The beam waist, i.e. the minimum beam radius at the focal position, is linearly scaled with $\sqrt{M^2}$ compared to a pure Gaussian beam, which has $M^2 = 1$. A small beam waist is particularly important for frequency conversion, where the efficiency scales in a non-linear fashion with the laser power density.

The reason for the decline in beam quality with increasing pump power is thermal lensing within the laser medium. It is a result of temperature induced transversal gradient of the crystal refractive index, mechanical stress, and deformation of the crystal surface, with the former effect being dominant. As the name suggests, thermal lensing leads to a focusing effect of the resonator mode; and when not incorporated in the resonator design can lead to reduced stability, power or beam quality. In order to quantify thermal lensing in the Mainz Ti:sapphire laser design, we used the approach described by Mirzaeian et al. [51]: The spatial evolution of a Gaussian beam can be described by ray-transfer matrices in the the so-called ABCD matrix formalism (see e.g. [52]), and therefore the measurement of beam radii at different positions z outside of the resonator can be used to extrapolate the beam properties back inside the resonator and compare it with the resonator design values. Repeating this approach for different pump powers can reveal information about the magnitude of thermal lensing. However, since Fig. 25 shows that higher transversal modes are involved, a hyper-Gaussian beam has to be considered and the M^2 factor has to be determined for each pump power setting. For the M^2-measurement, a standard Ti:sapphire cavity, operated near the gain maximum at 810 nm, was used. A plano-convex lens with $f = 150$ mm was placed at

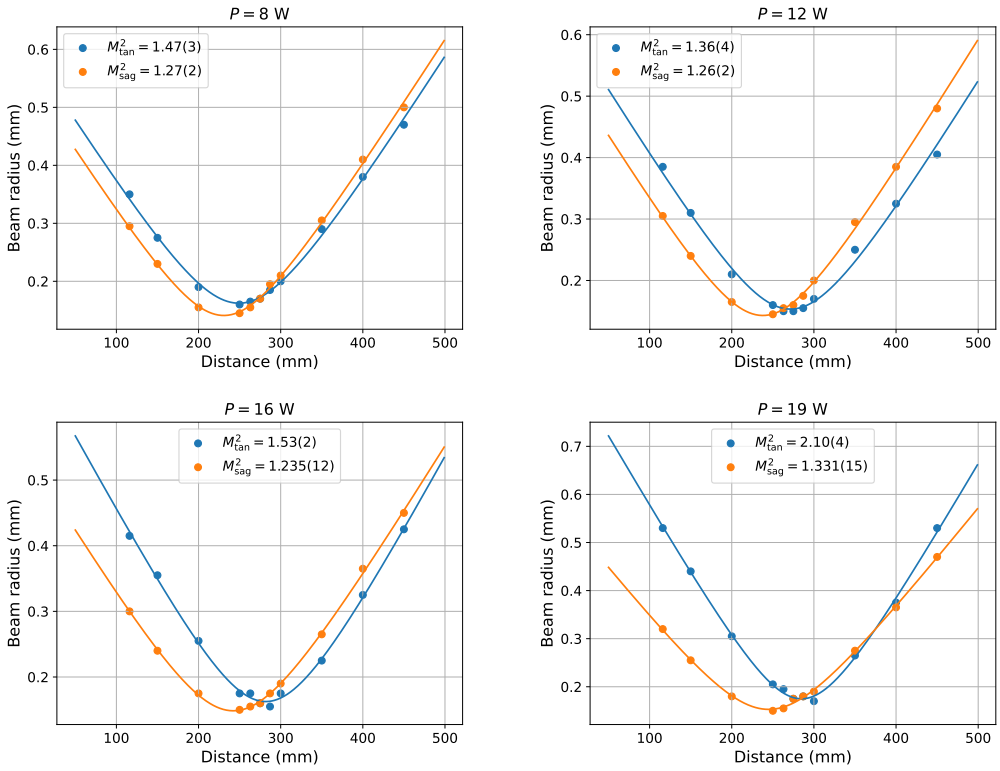


Figure 26: M^2 measurements for the standard Ti:sa laser at different pump powers. Tangential (horizontal) and sagittal (vertical) directions are treated separately. Distances are measured from the focusing lens.

a distance of 295 mm from the output coupler. A CMOS camera (Cinogy CinCam CMOS-1202) mounted on an optical rail was used to record the laser beam profile at variable distances from the lens across the focal point. The beam radius was determined according to the D4σ criterium using the included RayCi software. The M^2 factor was extracted by fitting

$$w(z) = w_0 \sqrt{1 + \left(\frac{(z - z(0)) M^2 \lambda}{n \pi w_0^2} \right)^2} \tag{15}$$

to the data, where z is the distance, w the beam radius, w_0 the beam waist, λ the wavelength and n the refractive index. The measurement was conducted independently in the tangential (horizontal) and sagittal (vertical) planes. The results of four different pump power settings are displayed in Fig. 26.

In a next step, the focal lens outside the resonator was removed. Beam radii were measured in a similar manner as described above, only that the beam is only diverging in this case. With the known resonator design parameters, i.e. distances, mirror radii of curvature (ROC) and intra-cavity elements, the beam radius within the resonator can be calculated using the ABCD matrix formalism and extrapolated to the location of the measured beam radii. For a fit to the data, only the thermal lensing parameter was varied, and M^2 values for the respective pump powers were fixed to the results from Fig. 26. The thermal lens was modeled simply by considering a thin lens in the center position of the Ti:sapphire crystal.

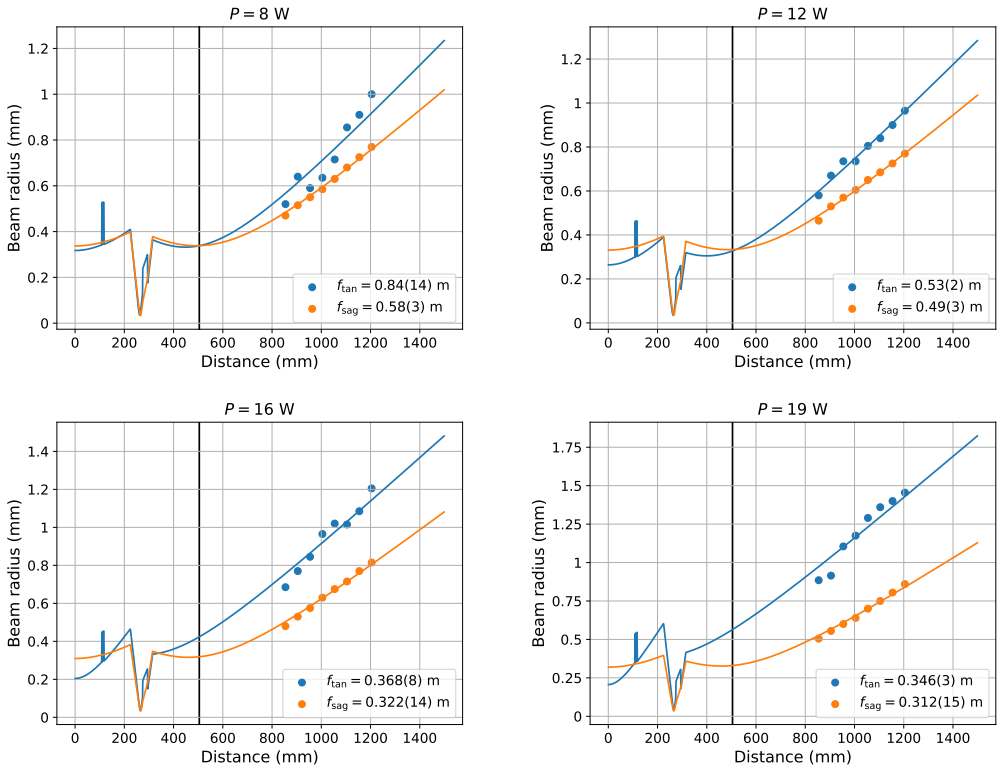


Figure 27: Thermal lensing measurement for the standard Ti:sapphire laser at different pump powers. The thermal lensing is modeled as thin lens in the middle of the Ti:sapphire crystal, with focal length f_{tan} in tangential (horizontal) direction and f_{sag} in sagittal (vertical) direction. The black vertical line indicates the laser output coupler.

The results are presented in Fig. 27. Some data points for the tangential direction show strong scattering around the fitted curve, most prominently visible in the 8 W and 19 W pump power cases. The significance of this effect is hard to assess, since the automated D4σ beam diameter determination of the RayCi software does not yield uncertainties. It is expected to be the result of a slight multimode structure of the beam, where a side peak is in some cases included in the D4σ criterium and sometimes not. Nonetheless, the extrapolation of the resonator beam radius can be fitted to the data and yields decreasing thermal focal lengths for increasing pump power, as expected. The given uncertainties are automatically generated by the fitting routine and most likely underestimate the error, because of the lack of uncertainties on the beam radii.
In addition to the results presented in Fig. 27, a different model of thermal lensing was tested. In this case the crystal is modeled as a waveguide with a quadratically varying refractive index in radial direction

$$n(r) = n_0(1 - \frac{1}{2}\gamma^2 r^2) \tag{16}$$

where n_0 is the refractive index on axis and γ represents the magnitude of thermal lensing. The corresponding ray transfer matrix is

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} \cos(\gamma l) & \gamma^{-1}\sin(\gamma l) \\ -\gamma \sin(\gamma l) & \cos(\gamma l) \end{pmatrix} \tag{17}$$

with the crystal length l [53]. This model showed a similar data representation as the thin lens model, with a somewhat less intuitive thermal lensing parameter. An even more sophisticated fitting model is given in [54]. Finally, the results of the thin lens model were considered for the following design steps.

The geometry of the new high-power Ti:sapphire laser was chosen to be very similar to the standard Ti:sapphire laser (see Sec. 2.2), except that the placement of two gain crystals in the central arm was considered. In this way the results from the thermal lensing measurements can directly be applied to the resonator calculation without the need for extrapolation, which may be unreliable. Using a Photonics DM60-532 laser with $M^2 = 15.8$ and a full angle far-field divergence of $V = 9.4\,\text{mrad}$ as pump source, a pump power of 16 W per crystal was chosen as design value, which is safely below the damage threshold of the crystal-air interface. The thermal lens focal lengths from the lower left panel of Fig. 27 were applied to each crystal and the new resonator was optimized for stability and astigmatism compensation by choosing appropriate distances and folding angle for the $r = 75\,\text{mm}$ curved mirrors in the central arm. A $f = 100\,\text{mm}$ lens was chosen to focus the pump beam into the crystal. It should be noted that, according to the design calculations, the resonator would become unstable without thermal lensing (i.e. $f_{\text{therm}} \to \infty$). The results of the resonator ray transfer matrix calculations are shown in Fig. 28 (First design). According to these specifications, a base plate for a laser prototype was manufactured. First tests at 10 kHz repetition rate with a pump power of 16 W per crystal showed promising results. An output power of 10.5 W could be achieved near 795 nm, effectively doubling the output power compared to the standard Ti:sapphire laser (cf. Tab. 3). Similar to the procedure described above, an M^2-measurement was performed. The results are shown in Fig. 29. In the tangential plane, the $M^2 = 1.62(8)$ value indicates a minor multimode structure, however, the astigmatism could be well compensated.

Eventually, during use of the new laser prototype, mirrors were repeatedly damaged due to the high intra-cavity power. Although this problem may be less relevant at 20 kHz repetition rate due to the lower pulse energy, a refined geometry with a larger intra-cavity beam radius was designed. The new design features a $r = 100\,\text{mm}$ radius of curvature of the concave mirrors, resulting in a longer cavity with a beam radius of $\approx 0.4\,\text{mm}$ on the mirror surfaces, compared to $\approx 0.3\,\text{mm}$ in the previous design. Calculated beam radii for this "Second design" are also included in Fig. 28. They key parameters for both designs are summarized in Table 10. An overview of the resonator layout of the second design is shown in Fig. 30. Similar to the standard Ti:sapphire laser, it also includes the option for intra-cavity second harmonic generation (IC-SHG). For this the output-coupler is replaced by a broadband high-reflective mirror (330 nm to 1100 nm), closing the cavity for the fundamental radiation. The second harmonic is coupled out using a longpass filter.

The laser prototype has been used several times in RISIKO standard operation at 10 kHz repetition rate. Unfortunately, it could not be thoroughly characterized until now due to a water chiller failure of the pump laser. Upon replacement a full characterization, including

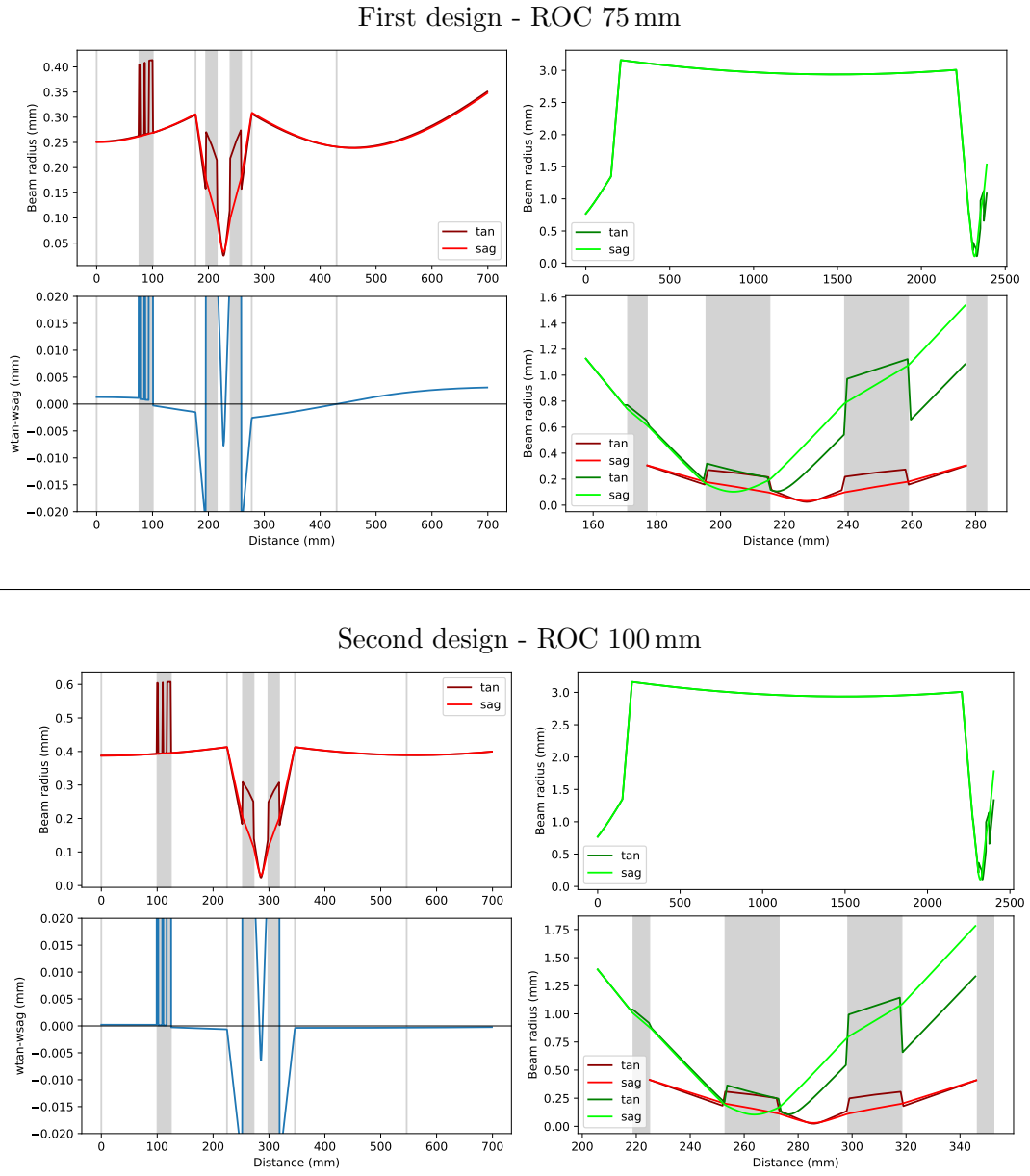


Figure 28: Beam radius calculation for the high-power Ti:sapphire laser design.
Upper left: Beam radius in tangential (dark red) and sagittal (light red) planes. Resonator elements are indicated by gray vertical lines. From left to right: End mirror, birefringent filter, curved mirror, Ti:sa crystal, Ti:sa crystal, curved mirror, output coupler. The beam is extrapolated beyond the output coupler.
Lower left: Beam radius difference between tangential and sagittal radii.
Upper right: Pump beam radius. The distance of zero corresponds to the pump laser exit. The beam is widened by a $f = -50\,\mathrm{mm}, f = 100\,\mathrm{mm}$ telescope and focused in the crystal with a $f = 100\,\mathrm{mm}$ lens at approximately 2 m distance.
Lower right: Detail view on the pump and resonator mode overlap. Distances given with respect to the end mirror (same as upper left panel). The vertical lines indicate intra-cavity elements. From left to right: Curved mirror, Ti:sa crystal, Ti:sa crystal, curved mirror. Only the pump beam injected from one side is displayed for clarity.

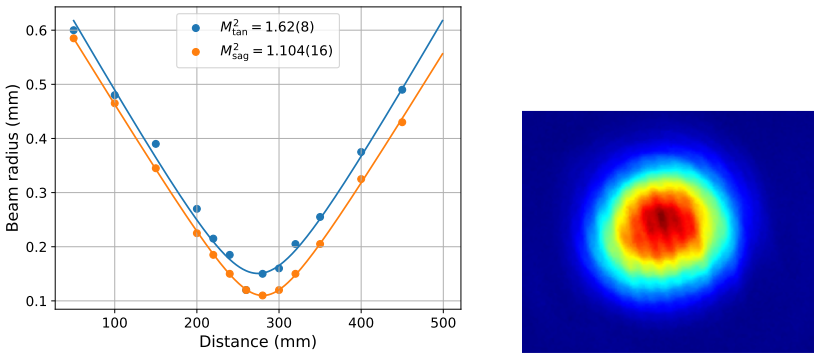


Figure 29: **Left:** M^2 measurement of the high power Ti:sapphire laser prototype (first design). **Right:** Single beam profile recording.

Parameter	First design	Second design
ROC (mm)	75	100
Fold angle (°)	21.5	19.0
OC/HR - CM (mm)	152	200
CM - Ti:sa (mm)	18.5	28.0
Ti:sa - Ti:sa (mm)	23.5	25.5

Table 10: Design parameters of the high power Ti:sapphire laser. The lower three parameters list the distances between the respective optical elements. Abbreviations: **ROC:** Concave mirror radius of curvature. **OC:** Output coupler. **HR:** High reflector. **CM:** Curved mirror. The first design features an overall smaller beam radius due to the shorter ROC and is therefore more likely sustain damage to the optics, whereas the second design is more prone to variations in thermal lensing.

20 kHz repetition rate tests, is planned.

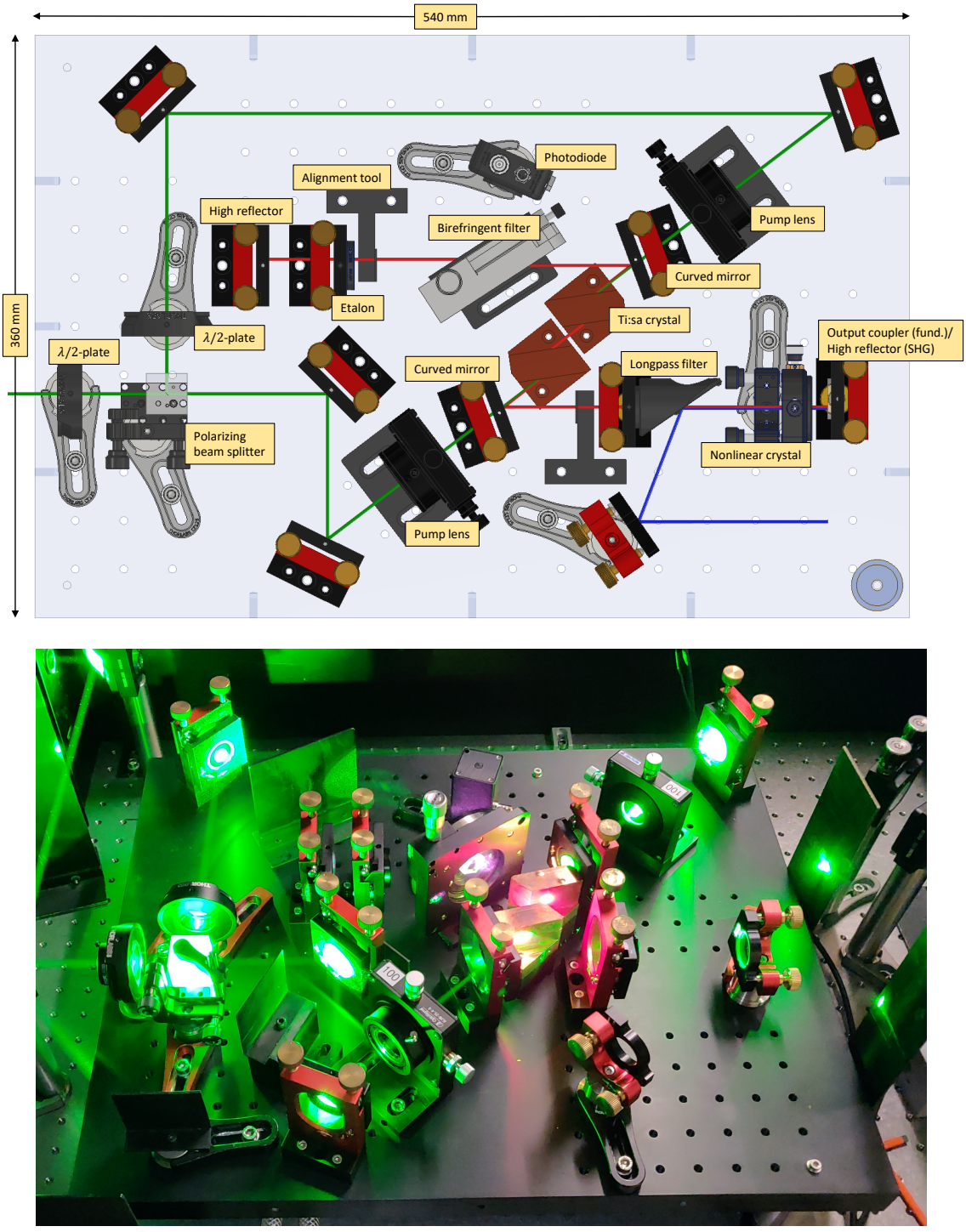


Figure 30: **Top:** Resonator layout of the high-power Ti:sapphire laser prototype, including the option for intra-cavity second harmonic generation (IC-SHG). For IC-SHG, the output coupler is replaced by a broadband high-reflecting mirror, closing the cavity for the fundamental radiation. The second harmonic radiation is coupled out with a longpass filter. **Bottom:** Photograph of the laser prototype (without IC-SHG).

6.2 Technical drawings

The supplemental material of this report includes the technical drawings of relevant ion source and ion optical components that can be used for the construction of a new mass separator. The vacuum chamber of the ion source unit is based in the ISOLDE target unit and compatible with an ISOLDE-type front-end. The ion source assembly consists of a separate sample reservoir and atomizer tube. It is optimized for off-line RILIS and rapid sample exchange. Alternative drawings for a larger diameter atomizer tube are included. The Einzel lens and deflector electrodes feature a clear aperture of 100 mm and are therefore suited for high-current beams with large diameters. An overview of the included parts up to the quadrupole triplet is given in Fig. 6.2. For the latter no 3D CAD drawing is available, but a sketch is given in Fig. 4. In addition to the parts shown in Fig. 6.2, some downstream components of the magnet are included, i.e. Faraday Cup, post-focalization Einzel lens, and implantation target. The magnet itself is not included, since it has to be replaced in the new separator design to a possibly up-scaled version. Downstream z-positions of individual components are summarized in Table 11.
The 3D CAD technical drawings are available as Autodesk Inventor 2022 files and as STEP (.stp) file for universal compatibility. They are hosted at a Seafile server of Rhineland Palatinate and can be accessed via https://seafile.rlp.net/d/cff49f16f96a4777950d/.

Table 11: Downstream positions of ion optical components. The downstream plane of the atomizer front panel marks the $z = 0$ position. The z-positions of all other components are given with respect the the entrance plane of the beam. Components market with † are given in z'-coordinates, i.e. measured from the magnet pole exit. Components marked with ‡ were not included in the IBSimu simulation. In this case their respective position at RISIKO is given.

Component	z-position (mm)
Atomizer panel	0
Heat shield	1
Extractor	40
Einzel lens	395
Deflectors	1095
Quadrupole triplet	2320
Magnet pole entrance	3090
Slit	$1080^{\dagger}$
Post focalization	$1680^{\dagger,\ddagger}$
Target	$2030^{\dagger,\ddagger}$

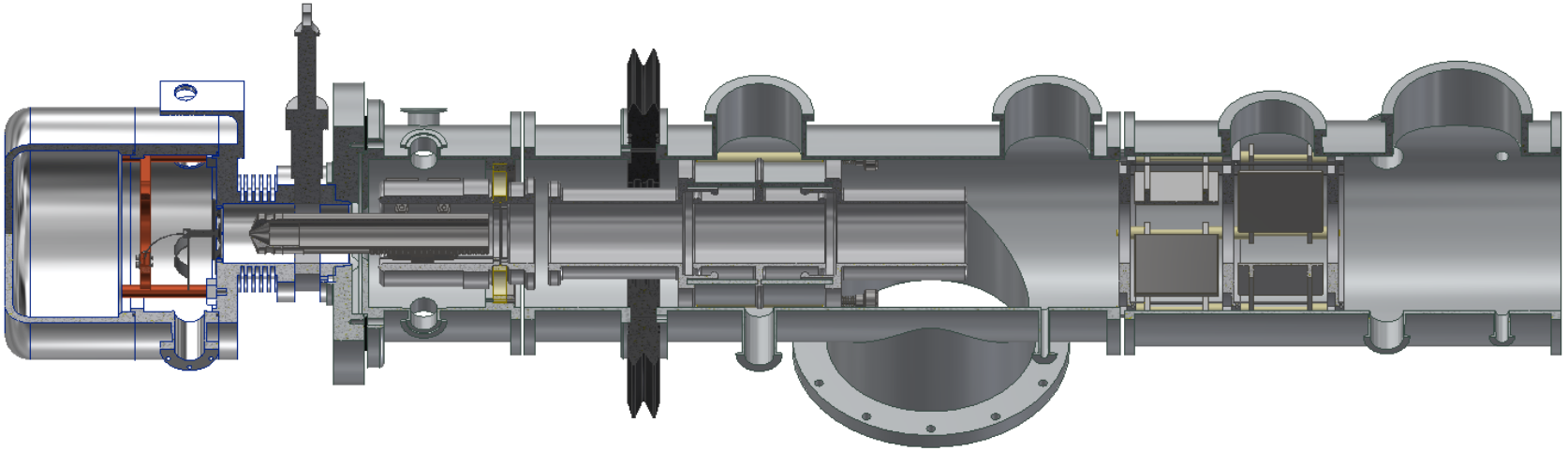


Figure 31: Overview of CAD parts as three-quarter section view of the beam line. The source region and ion optics up to the deflector plates is included. A CAD drawing of the quadrupole triplet is not available, but a drawing is included in Fig. 4. Distances between components are listed in Table 11.

6.3 Simulation Code

In order to reproduce or reevaluate the separator design, the IBSimu code is included in the following section. Simulation parameters can be changed as needed. The folder structure of the IBSimu simulation is depicted in Fig. 6.3. Individual files and their content is listed below, followed by a brief description of each file when needed. The code is commented thoroughly to enable changes by users other than the author. At the end of this section (Sec. 6.0.1), an installation guide for IBSimu, taken for the wiki of the LARISSA group, is included. 3D drawings of electrodes in the form of .stl files are required to run this simulation. These files are, together with the simulation code in digital form, are available at https://seafile.rlp.net/d/cff49f16f96a4777950d/. The .stl files are 3D meshes of individual electrodes and are based on a simplified version of the mass separator. Editing .stl files is not very practical and therefore the simplified mass separator drawings are also included as Autodesk Inventor 2022 files.

Listings

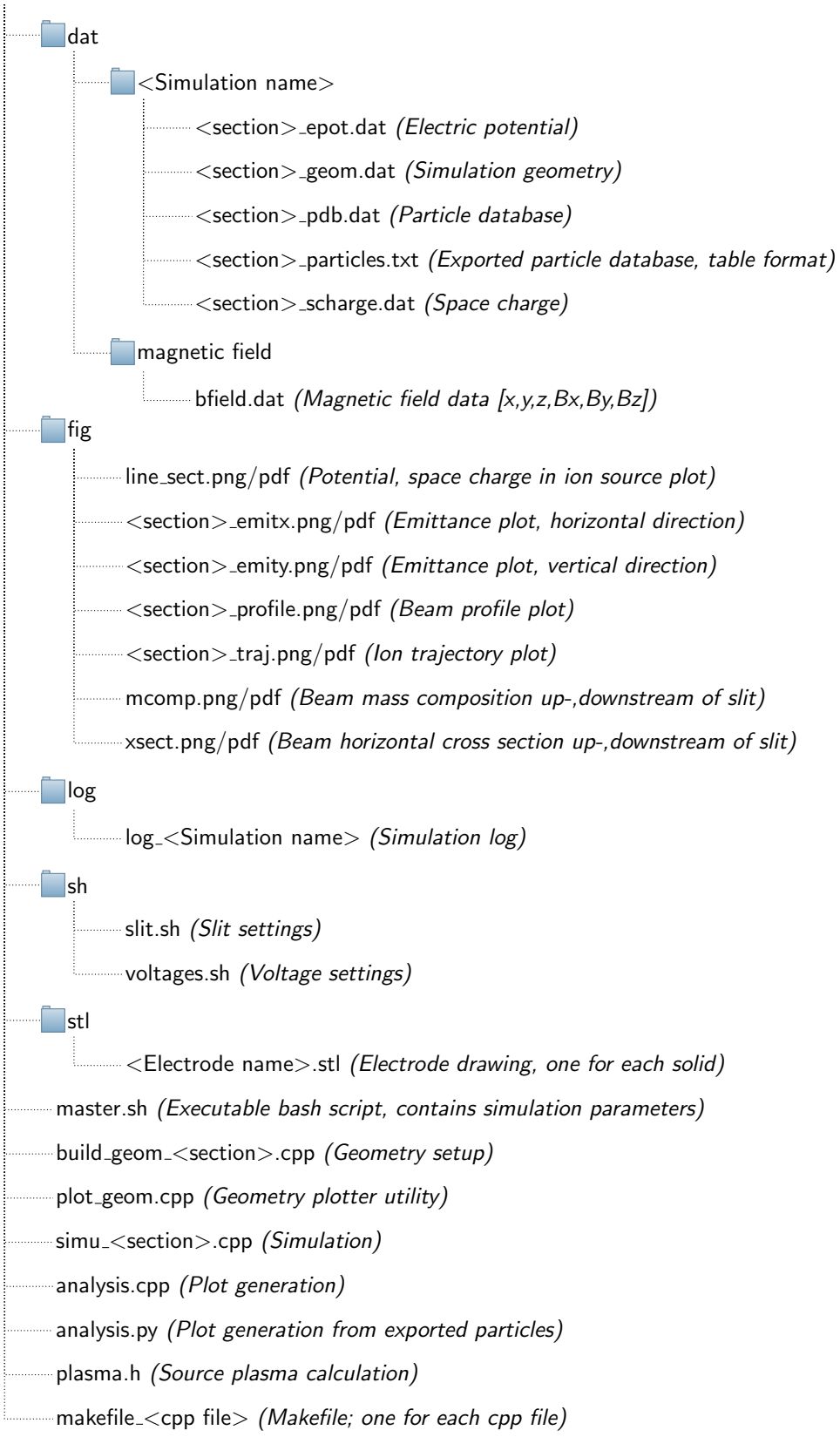


Figure 32: Folder structure of the IBSimu simulation.

The main executable is the bash script *master.sh*, where simulation parameters are set and all other simulation modules are called. Voltages and separator slit parameters are separately set in *sh/voltages.sh* and *sh/slit.sh*. The user can store multiple version of these parameter sets and reference to the desired file in *master.sh* (lines 116,117).
Each section of the separator (Extraction, ion optics, magnet, slit) has a separate C++ file to initialize the simulation geometry (*build_geom_<section>.cpp*) and perform the simulation (*simu_<section>.cpp*). Plots are generated using *analysis.cpp*. Each of these modules are called within *master.sh* and are supplied with command line arguments (lines 126-133). Individual simulation sections can be omitted by commenting the respective lines, however, each section of the separator needs an existing particle database exported from the previous section. Obviously, geometries have to be re-built when changing the simulation geometry of the respective section, but also when voltages are changed. Prior to simulation, it is recommended to check the geometry by commenting all *simu_<section>* and *analysis* calls in lines 126-133 and uncommenting line 123 with the respective argument for the section geometry to be displayed: *ex, io, mag, sl* for extraction, ion optics, magnet and slit. Finally, in line 134, *analysis.py* is called to generate the plots similar to Fig. 15 from the particle data of the slit simulation.

Listing 1: master.sh

```
1  # geom params (general)
2  stl_path="stl/" # stl import path; REMEMBER TO ADJUST LINE PARAMS ACCORDINGLY (below AND in
       build_geom_ex "inside" method)
3
4  # geom params (ex - line & extraction)
5  h_ex=0.227e-3 # mesh cell size in m
6  xsize_ex=40.86e-3 # full mesh size in x dir in m
7  ysize_ex=20.43e-3 # full mesh size in y dir in m
8  zmin_ex=-35.0e-3 # minimum of simuation area in z dir in m (x,y are set automatically)
9  zmax_ex=80.0e-3 # maximum of simuation area in z dir in m
10
11 # geom params (io - ion optics)
12 h_io=3.0e-3 # mesh cell size in m
13 xsize_io=105.0e-3 # full mesh size in x dir in m
14 ysize_io=52.5e-3 # full mesh size in y dir in m
15 zmin_io=40.0e-3 # minimum of simuation area in z dir in m (x,y are set automatically)
16 zmax_io=3010.0e-3 #  maximum of simuation area in z dir in m
17
18 # geom params (mag - magnet)
19 h_mag=3.0e-3 # mesh cell size in m
20 mag_zero=3090 #zero coordinate of magnet assembly in mm (NEEDS TO BE ADJUSTED WHEN USING
       TRANSLATE)
21 exit_x=-375.0 #pole (not pipe!) exit coordinates in mm
22 exit_z=849.52
23 # slit dist and separation are defined in slit.sh
24 defl=60 # deflection angle in deg
25 xsize_mag=2300.0e-3 #$((1175+105+$slit_dist-20))e-3 # include until 30mm before slit exit
       of mag: -1175; positive region in x dir 105 (ADJUST TO EXCLUDE SLITS!)
26 ysize_mag=54.0e-3 # full mesh size in y (NO MIRROR SYMMETRY)
27 zmin_mag=3000.0e-3 # 2300 to include Q triplet
28 zmax_mag=5100.0e-3 #  maximum of simuation area in z dir in m
29 bfn="dat/magnetic_field/bfield.dat" # bfield filename
30 # magnetic field scaling defined in slit.sh
```

```
31
32 # geom params (slit - sl)
33 h_sl=0.25e-3 # mesh cell size in m
34 xsize_sl=40.0e-3 # full mesh size in x dir in m
35 ysize_sl=20.0e-3 # full mesh size in y dir in m
36 # z size is defined is slit.sh
37
38 # simu params (general)
39 it=5 # number of Vlasov iterations (default 15, but converges usually faster)
40 it_ex=5 # for extraction
41 tol=1e-5 #electric field solver uncertainty tolerance (default 1e-4)
42 sc_alpha=0.8 # space charge averaging parameter e(0,1). small values for high dampening.
       rho = alpha*rho_new + (1-alpha)*rho_old
43
44 # simu params (ex)
45 r_line=1.25e-3 # line inner radius in m (CHANGE ALSO IN build_geom_ex)
46 h_line=0.0e-3 # line height in m (consider slot hole with length h_line and radius r_line;
       set 0 for circular)
47 l_line=35.0e-3 # beam start in z direction in m (0=panel exit)
48 #line params hardcoded in build_geom_ex("inside" function) :adjust accordingly
49 eps=0.3e-3 # distance of LASER ION particle start position from line wall
50 # plasma parameters
51 use_ibsimu_plasma=1 # use pext plasma model from ibsimu (1)
52 use_analytic_plasma=0 # implements analytic potential within line according to Lawson CERN
       report 76-09
53 show_surface_ions=1 # whether to generate tracetories or set them as collided after
       generation
54 z_plasma=-0.0e-3 # plasma at z < zplasma in m
55 W=4.19 # atomizer material work function in eV (only used by analytic plasma model)
56 Vplasma=-2.52 # Plasma potential in V  (only used by IBsimus plasma model)
57 Te=2300 # electron temperature/thermal energy (in K); (only used by IBsimu plasma model;
       IBsimu uses eV for this value, get converted later)
58 Ti=$Te #$Te # surface ion temperature in K, (used by both plasma models) (1000K=0.086eV)
59 T0=$Te # laser ion temperature in K
60 U_line=2.5 # line voltage in V for additional particle velocity in z direction (laser ions
       only)
61 Ni=5.0e4 #2 number surface ion macro particles
62 N0=2.0e5 # number of laser ion macro  particles
63 massi=40.0 # microscopic surface ion mass in u (can be set very high to minimize movement
       of surface ion --> quasi-stationary)
64 mass0=150.0 # microscopic particle mass in u
65 ni=2e12 #bulk plasma (surface) ion density in e/cm^-3 (used by both plasma models)
66 #I0=1e-9 # laser ion current in A; NOT USED, DEFINED AS I BELOW
67
68 # analysis params (general)
69 interact=0 # 0 for geomplotter (static) and 1 for gtkplotter (interactive)
70 view="zx" # select axes (zx, zy or xy); GeomPlotter only
71 offs=0.0e-3 # offset along the third axis, i.e. the axis which is not included in "view";
       GeomPlotter only
72 Jview=1 # if 1, display current density (CHANGED TO SPACE CHARGE-BETTER INTERPOLATED)
       instead of particle trajectories; GeomPlotter only
73 usrlim=0 # if 0 plot full range, if 1 use user defined limits
74 zmin_plt=-35.0e-3 # plot axis limits in m; GeomPlotter only
75 zmax_plt=60.0e-3
76 xymin_plt=0
77 xymax_plt=10.0e-3
```

```
diag_ax_ex="z" # x or z
diag_ax_io="z"
diag_ax_mag="x"
diag_ax_sl="z"
diag_ex=79.0e-3 # coordinate along specified axis which beam diagnostics are perforemed
diag_io=3000.0e-3 # coordinate along z axis which beam diagnostics are perforemed
diag_mag=-700.0e-3 # x threshold below which particles get exported to slit
# diag sl in slit.sh
efmt="png" # export figure format (png, pdf or svg)

fn="sanda" # output filename
gfn="d25_"$fn # geometry filename suffix (e.g. gfn=test --> dat/geom_test.dat)
lfn="log/log_"$gfn".txt" # full log filename (logs stdout)

# ##################################### INIT ###################################
starttime=$(date +%d-%b-%H:%M)
printf "start time: "$starttime"\n" 2>&1 | tee $lfn # create logfile and print date (overwrite previous file with same name)

printf "\nRunning makefiles\n" 2>&1 | tee -a $lfn
make -f makefile_geom_ex 2>&1 | tee -a $lfn
make -f makefile_geom_io 2>&1 | tee -a $lfn
make -f makefile_geom_mag 2>&1 | tee -a $lfn
make -f makefile_geom_sl 2>&1 | tee -a $lfn
make -f makefile_plt_geom 2>&1 | tee -a $lfn
make -f makefile_simu_ex 2>&1 | tee -a $lfn
make -f makefile_simu_io 2>&1 | tee -a $lfn
make -f makefile_simu_mag 2>&1 | tee -a $lfn
make -f makefile_simu_sl 2>&1 | tee -a $lfn
make -f makefile_analysis 2>&1 | tee -a $lfn
sleep 3 # seconds wait the the user can see errors occurring in makefiles :)

# ##################################### DO STUFF ###############################
mkdir "dat/"$gfn # generate directories if not already existing
mkdir "fig/"$gfn

for I in 100e-9 #1000e-9 10000e-9 # for loop for different currents
do

. ./sh/voltages_25_focus.sh # voltages saved in other bash script
. ./sh/slit_25.sh # slitt settings in other bash script
mag_zero=$(($mag_zero+$translate))
./build_geom_ex $gfn $stl_path $h_ex $xsize_ex $ysize_ex $zmin_ex $zmax_ex $Vsrc $Vex $translate 2>&1 | tee -a $lfn # construct geometry
./build_geom_io $gfn $stl_path $h_io $xsize_io $ysize_io $zmin_io $zmax_io $Vsrc $Vex $Vbias $Veinzel $Vdefx $Vdefy $Vq13x $Vq13y $Vq2x $Vq2y $translate 2>&1 | tee -a $lfn # construct geometry
./build_geom_mag $gfn $stl_path $h_mag $xsize_mag $ysize_mag $zmin_mag $zmax_mag $Vsrc $translate $defl $slit_dist $slit_sep 2>&1 | tee -a $lfn # construct geometry
./build_geom_sl $gfn $stl_path $h_sl $xsize_sl $ysize_sl $zmin_sl $zmax_sl $Vsrc $mag_zero $exit_x $exit_z $slit_dist $slit_sep $defl 2>&1 | tee -a $lfn # construct geometry
#./plot_geom "ex" $gfn # plot geometry (without simulation)

sfn=$fn"_I"$I # simulation filename suffix (e.g. sfn=test --> dat/epot_test.dat, fig/fig_test.png, ...)
./simu_ex $gfn $sfn $r_line $h_line $l_line $eps $zmax_ex $use_ibsimu_plasma
```

```
    $use_analytic_plasma $show_surface_ions $z_plasma $W $Vplasma $Te $Ti $T0 $U_line $Ni
    $N0 $massi $mass0 $ni $I $it_ex $tol $sc_alpha 2>&1 | tee -a $lfn # simulate
    trajectories
./analysis "ex" $gfn $sfn $interact $view $offs $Jview $usrlim $zmin_plt $zmax_plt
    $xymin_plt $xymax_plt $diag_ax_ex $diag_ex $r_line $efmt 2>&1 | tee -a $lfn # make plots
./simu_io $gfn $sfn $zmax_io $it $tol $sc_alpha 2>&1 | tee -a $lfn # simulate trajectories
./analysis "io" $gfn $sfn $interact $view $offs $Jview $usrlim $zmin_plt $zmax_plt
    $xymin_plt $xymax_plt $diag_ax_io $diag_io $r_line $efmt 2>&1 | tee -a $lfn # make plots
./simu_mag $gfn $sfn $bfn $bscale $diag_mag $mag_zero $it $tol $sc_alpha 2>&1 | tee -a $lfn
    # simulate trajectories
./analysis "mag" $gfn $sfn $interact $view $offs $Jview $usrlim $zmin_plt $zmax_plt
    $xymin_plt $xymax_plt $diag_ax_mag $diag_mag $r_line $efmt $bfn $bscale $mag_zero 2>&1
    | tee -a $lfn # make plots
./simu_sl $gfn $sfn $mag_zero $exit_x $exit_z $slit_dist $slit_sep $defl $diag_sl $filter
    $it $tol $sc_alpha 2>&1 | tee -a $lfn # simulate trajectories
./analysis "sl" $gfn $sfn $interact $view $offs $Jview $usrlim $zmin_plt $zmax_plt
    $xymin_plt $xymax_plt $diag_ax_sl $diag_sl $r_line $efmt 2>&1 | tee -a $lfn # make plots
python3 analysis.py $fn $gfn $I $slit_dist $slit_sep $bscale | tee -a $lfn

printf "END OF LOOP I="$I" ##################################################\n" 2>&1 | tee
    -a $lfn
date

done # end of I loop

# ##################################### END #####################################
# print start and end time
endtime=$(date +%d-%b-%H:%M)
printf "\nstart time: "$starttime"\n" 2>&1 | tee -a $lfn
printf "end time: "$endtime"\n" 2>&1 | tee -a $lfn
```

Voltages to apply to the electrodes are stored in *voltages.sh*. Different voltage settings can be stored in the *sh* folder and the filename to be used is specified in *master.sh* (line 116). Additionally, voltages of one set can vary based on the ion beam current (*if*-structre in lines 1, 14, 27, 40).

Listing 2: voltages.sh

```
if [[ $I = 100e-9 ]]
then
translate=0 # extractor(and downstream) translation in mm, 0=40mm (NOT TESTED!)
Vsrc=30.0e3 # source voltage in V
Vex=20.0e3 # extractor voltage in V
Vbias=0.0 # Veinzel13 in V
Veinzel=9.8e3 # Veinzel2 in V (parallel beam @ 10)
Vdefx=0.0 # x deflector in V
Vdefy=0.0 # y deflector in V (Y DEFLECTOR NOT WORKING PROPERLY BECAUSE OF SYMMETRIC
    POTENTIAL!)
Vq13x=0.0 #-310.0 # Q triplet 1,3 horizontal in V
Vq13y=0.0 #+310.0 # Q triplet 1,3 vertical in V
Vq2x=0.0 #+350.0 # Q triplet 2 horizontal in V
Vq2y=0.0 #-350.0 # Q triplet 2 vertical in V
elif [[ $I = 1000e-9 ]]
then
translate=0 # extractor(and downstream) translation in mm, 0=40mm
```

```
17 Vsrc=30.0e3 # source voltage in V
18 Vex=20.0e3 # extractor voltage in V
19 Vbias=0.0 # Veinzel13 in V
20 Veinzel=10.25e3 # Veinzel2 in V (parallel beam @ 11.4)
21 Vdefx=0.0 # x deflector in V
22 Vdefy=0.0 # y deflector in V (Y DEFLECTOR NOT WORKING PROPERLY BECAUSE OF SYMMETRIC
       POTENTIAL!)
23 Vq13x=0.0 #-125.0 # Q triplet 1,3 horizontal in V
24 Vq13y=0.0 #125.0 # Q triplet 1,3 vertical in V
25 Vq2x=0.0 #140.0 # Q triplet 2 horizontal in V
26 Vq2y=0.0 #-140.0 # Q triplet 2 vertical in V
27 elif [[ $I = 10000e-9 ]]
28 then
29 translate=0 # extractor(and downstream) translation in mm, 0=40mm
30 Vsrc=30.0e3 # source voltage in V
31 Vex=20.0e3 # extractor voltage in V
32 Vbias=0.0 # Veinzel13 in V
33 Veinzel=12.5e3 # Veinzel2 in V
34 Vdefx=0.0 # x deflector in V
35 Vdefy=0.0 # y deflector in V (Y DEFLECTOR NOT WORKING PROPERLY BECAUSE OF SYMMETRIC
       POTENTIAL!)
36 Vq13x=+710.0 # Q triplet 1,3 horizontal in V
37 Vq13y=-710.0 # Q triplet 1,3 vertical in V
38 Vq2x=-770.0 # Q triplet 2 horizontal in V
39 Vq2y=+770.0 # Q triplet 2 vertical in V
40 else # default settings
41 translate=0 # extractor(and downstream) translation in mm, 0=40mm
42 Vsrc=30.0e3 # source voltage in V
43 Vex=20.0e3 # extractor voltage in V
44 Vbias=0.0 # Veinzel13 in V
45 Veinzel=10.0e3 # Veinzel2 in V
46 Vdefx=0.0 # x deflector in V
47 Vdefy=0.0 # y deflector in V (Y DEFLECTOR NOT WORKING PROPERLY BECAUSE OF SYMMETRIC
       POTENTIAL!)
48 Vq13x=0 #-1200.0 # Q triplet 1,3 horizontal in V
49 Vq13y=0 #+1200.0 # Q triplet 1,3 vertical in V
50 Vq2x=0 #+1100.0 # Q triplet 2 horizontal in V
51 Vq2y=0 #-1100.0 # Q triplet 2 vertical in V
52 fi
```

Slit settings, i.e. distance and separation of the vertical slit aperture ,can be set in *slit.sh*. Different slit settings can be stored in the *sh* folder and the filename to be used is specified in *master.sh* (line 117). Additionally, slit settings of one set can vary based on the ion beam current (*if*-structre in lines 1, 6, 13, 16). However, in the listing given here the individual settings are overwritten at lines 22 onward to provide a shard setting independent of ion current. The *bscale* parameter scales the imported magnetic field values, which are interpreted as Tesla.

Listing 3: slit.sh

```
1 if [[ $I = 100e-9 ]]
2 then
3 slit_dist=1085 # in mm from pole exit
4 slit_sep=2.0 #0.5 # in mm
```

```
bscale=0.02731 # scale magn field (SMALLER VALUE --> deflection to positive x)
elif [[ $I = 1000e-9 ]]
then
slit_dist=1015 # in mm from pole exit
slit_sep=2.5 #0.5 # in mm
bscale=0.027315 # scale magn field
elif [[ $I = 10000e-9 ]]
then
slit_dist=1602 # in mm from pole exit
slit_sep=3.5 #0.5 # in mm
bscale=0.027295 # 0.02731  scale magn field
else # default settings
slit_dist=1000 # in mm from pole exit
slit_sep=2.0 #0.5 # in mm
bscale=0.02731 # scale magn field
fi

# overwrite individual settings:
slit_dist=1085 # in mm from pole exit
bscale=0.02731 # scale magn field (SMALLER VALUE --> deflection to positive x)

filter=0 # if not 0, import only ions of given mass to  slit simu
zmin_sl=$(($slit_dist -21))e-3 #
zmax_sl=$(($slit_dist +10))e-3 # 25.0e-3 #  maximum of simuation area in z dir in m
diag_sl=$(($slit_dist-1))e-3 # 1mm in front of slit

# use the following for interactive focus search (and interact=1 in master.sh)
#slit_dist=300
#slit_sep=30
#h_sl=2.0e-3
#zmin_sl=$(($slit_dist -21))e-3 #
#zmax_sl=2000e-3
#interact=1
#filter=150
```

The following C++ source code files (all files with the .cpp file extension) have to be compiled prior to use. This is achieved by using makefiles. The listing below gives an example of one makefile, where *cppFile* is a placeholder for the filename to be compiled. The *master.sh* script runs all required makefiles (lines 97-106). Therefore, all makefiles listed in these lines should be present. In addition to this generic makefile, all specific makefiles for each .cpp file are stored at https://seafile.rlp.net/d/cff49f16f96a4777950d/.

Listing 4: makefile_<cpp file>

```
CC = g++
LIBS = `pkg-config --libs ibsimu-1.0.6`
LDFLAGS = `pkg-config --libs ibsimu-1.0.6`
CXXFLAGS = -Wall -std=c++11 -g `pkg-config --cflags ibsimu-1.0.6`

cppFile: cppFile.o
   $(CC) -o cppFile cppFile.o $(LDFLAGS)
cppFile.o: cppFile.cpp

clean:
```

```
11    $(RM) *.o cppFile
```

The following *build_geom_<section>.cpp* files intitalize the simulation mesh based on imported CAD drawings in the .stl format. One or more STL files can form a solid body in the simulation geometry. Afterwards, a mesh of solid, near-solid and vacuum nodes is constructed. Boundary conditions are set to each boundary of the geometry (Bounds 1-6 as $[x_0, x_{\max}, y_0, y_{\max}, z_0, z_{\max}]$) and to each solid (Bounds 7+). In case of *build_geom_ex.cpp*, the atomizer is constructed using a boolean function rather than a STL file. In this case the geometric atomizer parameters have to be set in the *inside*-method (lines 20-22). All *build_geom_<section>.cpp* files are similar in structure, but are all fully included here to readily adapt the simulation.

Listing 5: build_geom_ex.cpp

```
1  #include <fstream>
2  #include <iomanip>
3  #include <limits>
4  #include "dxf_solid.hpp"
5  #include "stl_solid.hpp"
6  #include "stlfile.hpp"
7  #include "mydxffile.hpp"
8  #include "geometry.hpp"
9  #include "func_solid.hpp"
10 #include "error.hpp"
11 #include "ibsimu.hpp"
12
13 using namespace std;
14
15
16 bool inside (double x, double y, double z){
17         // returns true if given tuple of coordinates (x,y,z) lies within line volume
18         // slot hole with radius r, height h and length(depth) l
19         // with 'eps' distance to walls/exit
20         double r = 1.25e-3; // #####
21         double h = 0.0e-3;
22         double l = 35.0e-3;
23         double eps = 0.0e-3;
24         return !(    ( x*x + (y-h/2)*(y-h/2) < (r-eps)*(r-eps) ) || // upper circle (x,y)
25                     ( y < h/2-eps && abs(x) < r-eps )   )   // slot (x,y)
26                 &&  ( z < 0.0-eps && z > -l+eps );  // z
27 }
28
29 void build_geom( int argc, char **argv ){
30
31     // get command line arguments
32     std::string gfn = argv[1]; // mesh export filename
33     std::string stl_path = argv[2]; // stl import path
34     const double h = atof(argv[3]); // mesh cell size
35     const double xsize = atof(argv[4]); // full mesh size in x dir in m
36     const double ysize = atof(argv[5]); // full mesh size in y dir in m
37     const double zmin = atof(argv[6]); // full mesh size in z dir in m
38     const double zmax = atof(argv[7]); // minimum of mesh area in z dir in m (x,y are set
       automatically)
```

```
39      const double Vsource = atof(argv[8]); // the script will shift this to 0 later and
        other voltages accordingly
40      const double Vex = atof(argv[9]); // extractor
41      const double translate = atof(argv[10]); // extractor(and downstream) elements
        translation
42
43      std::string geom_fn = "dat/" + gfn + "/ex_geom_" + gfn + ".dat"; // folder is named for
        geometry prefix
44
45      double sizereq[3] = { xsize,  // x full
46                            ysize,  // y half (mirror symmetry)
47                            zmax - zmin}; // z full
48      Int3D meshsize( (int)floor(sizereq[0]/h+1),
49                      (int)floor(sizereq[1]/h+1),
50                      (int)floor(sizereq[2]/h+1) );
51      Vec3D origo( -sizereq[0]/2, 0, zmin ); // z component in negative direction to include
        line
52      Geometry geom( MODE_3D, meshsize, origo, h );
53
54      Transformation T0; // applies to source
55      T0.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor export
        settings
56
57      Transformation T1; // applies to ex and further downstream
58      T1.translate( Vec3D( 0.0, 0.0, translate ) );
59      T1.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor export
        settings
60
61      //STLFile *fline = new STLFile( stl_path + "sanda_line.stl" );
62      //STLFile *fpanel = new STLFile( stl_path + "sanda_panel.stl" );
63      STLFile *fhs = new STLFile( stl_path + "sanda_pierce.stl" );
64      STLFile *fex = new STLFile( stl_path + "sanda_ex.stl" );
65
66      // line is not taken from drawings, but from function "inside", which makes everything
        at z<0 solid except for line interior
67      Solid *line = new FuncSolid( inside );
68      geom.set_solid( 7, line );
69
70      STLSolid *source = new STLSolid;
71      source->set_transformation( T0 );
72      //source->add_stl_file( fline );
73      //source->add_stl_file( fpanel );
74      source->add_stl_file( fhs );
75      geom.set_solid( 8, source );
76
77      STLSolid *ex = new STLSolid;
78      ex->set_transformation( T1 );
79      ex->add_stl_file( fex );
80      geom.set_solid( 9, ex );
81
82      geom.set_boundary(  1,  Bound(BOUND_NEUMANN,     0.0) ); // x0
83      geom.set_boundary(  2,  Bound(BOUND_NEUMANN,     0.0) ); // x_max
84      geom.set_boundary(  3,  Bound(BOUND_NEUMANN,     0.0) ); // y_0
85      geom.set_boundary(  4,  Bound(BOUND_NEUMANN,     0.0) ); // y_max
86      geom.set_boundary(  5,  Bound(BOUND_NEUMANN,   0.0) ); // z_0 # neumann 0 for positive
        plasma model
```

```
    geom.set_boundary(  6,  Bound(BOUND_DIRICHLET,  Vex - Vsource) ); // z_max

    geom.set_boundary(  7,  Bound(BOUND_DIRICHLET,  0.0) ); // line
    geom.set_boundary(  8,  Bound(BOUND_DIRICHLET,  0.0) ); // source (heatshield, ...)
    geom.set_boundary(  9,  Bound(BOUND_DIRICHLET,  Vex - Vsource) ); // ex

    geom.build_mesh();
    geom.save( geom_fn );

}


int main( int argc, char **argv )
{
    try {
   //ibsimu.set_message_output( "ibsimu.txt" );
        ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
       ibsimu.set_thread_count( 4 );
       build_geom( argc, argv );
    } catch( Error e ) {
       e.print_error_message( ibsimu.message( 0 ) );
        exit( 1 );
    }

    return( 0 );
}
```

Listing 6: build_geom_io.cpp

```
#include <fstream>
#include <iomanip>
#include <limits>
#include "dxf_solid.hpp"
#include "stl_solid.hpp"
#include "stlfile.hpp"
#include "mydxffile.hpp"
#include "geometry.hpp"
#include "func_solid.hpp"
#include "error.hpp"
#include "ibsimu.hpp"

using namespace std;

void build_geom( int argc, char **argv ){

    // get command line arguments
    std::string gfn = argv[1]; // mesh export filename
    std::string stl_path = argv[2]; // stl import path
    const double h = atof(argv[3]); // mesh cell size
    const double xsize = atof(argv[4]); // full mesh size in x dir in m
    const double ysize = atof(argv[5]); // full mesh size in y dir in m
    const double zmin = atof(argv[6]); // full mesh size in z dir in m
    const double zmax = atof(argv[7]); // minimum of mesh area in z dir in m (x,y are set
    automatically)
    const double Vsource = atof(argv[8]); // the script will shift this to 0 later and
```

```
    other voltages accordingly
26  const double Vex = atof(argv[9]); // extractor
27  const double Vbias = atof(argv[10]); // = Veinzel13#
28  const double Veinzel = atof(argv[11]); // einzel mid electrode
29  const double Vdefx = atof(argv[12]);
30  const double Vdefy = atof(argv[13]);
31  const double Vq13x = atof(argv[14]); // Q triplet
32  const double Vq13y = atof(argv[15]);
33  const double Vq2x = atof(argv[16]);
34  const double Vq2y = atof(argv[17]);
35  const double translate = atof(argv[18]); // extractor(and downstream) elements
    translation
36
37  std::string geom_fn = "dat/" + gfn + "/io_geom_" + gfn + ".dat"; // folder is named for
    geometry prefix
38
39  double sizereq[3] = { xsize, // x full
40                        ysize, // y half (mirror symmetry)
41                        zmax - zmin}; // z full
42  Int3D meshsize( (int)floor(sizereq[0]/h)+1,
43                  (int)floor(sizereq[1]/h)+1,
44                  (int)floor(sizereq[2]/h)+1 );
45  Vec3D origo( -sizereq[0]/2, 0, zmin );
46  Geometry geom( MODE_3D, meshsize, origo, h );
47
48  Transformation T0; // applies to source
49  T0.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor export
    settings
50
51  Transformation T1; // applies ex and further downstream
52  T1.translate( Vec3D( 0.0, 0.0, translate ) );
53  T1.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor export
    settings
54
55  STLFile *fex = new STLFile( stl_path + "sanda_ex.stl" );
56  STLFile *fex_holder = new STLFile( stl_path + "sanda_ex_holder.stl" );
57  STLFile *fel1 = new STLFile( stl_path + "sanda_einzel1.stl" );
58  STLFile *fel2 = new STLFile( stl_path + "sanda_einzel2.stl" );
59  STLFile *fel3 = new STLFile( stl_path + "sanda_einzel3.stl" );
60  STLFile *fdefl_yu = new STLFile( stl_path + "sanda_defl_yu.stl" ); // y upper
61  STLFile *fdefl_yd = new STLFile( stl_path + "sanda_defl_yd.stl" ); // y lower
62  STLFile *fdefl_xl = new STLFile( stl_path + "sanda_defl_xl.stl" ); // x left
63  STLFile *fdefl_xr = new STLFile( stl_path + "sanda_defl_xr.stl" ); // x right
64  STLFile *fQ13_y = new STLFile( stl_path + "sanda_Q13y.stl" ); // Q triplet 1,3 vertical
65  STLFile *fQ13_x = new STLFile( stl_path + "sanda_Q13x.stl" ); // Q triplet 1,3
    horizontal
66  STLFile *fQ2_y = new STLFile( stl_path + "sanda_Q2y.stl" ); // Q triplet 2 vertical
67  STLFile *fQ2_x = new STLFile( stl_path + "sanda_Q2x.stl" ); // Q triplet 2 horizontal
68
69  STLSolid *ex = new STLSolid;
70  ex->set_transformation( T1 );
71  ex->add_stl_file( fex );
72  ex->add_stl_file( fex_holder );
73  geom.set_solid( 7, ex );
74
75  STLSolid *el1 = new STLSolid;
```

```
76      el1->set_transformation( T1 );
77      el1->add_stl_file( fel1 );
78      geom.set_solid( 8, el1 );
79
80      STLSolid *el2 = new STLSolid;
81      el2->set_transformation( T1 );
82      el2->add_stl_file( fel2 );
83      geom.set_solid( 9, el2 );
84
85      STLSolid *el3 = new STLSolid;
86      el3->set_transformation( T1 );
87      el3->add_stl_file( fel3 );
88      geom.set_solid( 10, el3 );
89
90      STLSolid *defl_yu = new STLSolid;
91      defl_yu->set_transformation( T1 );
92      defl_yu->add_stl_file( fdefl_yu );
93      geom.set_solid( 11, defl_yu );
94
95      STLSolid *defl_yd = new STLSolid;
96      defl_yd->set_transformation( T1 );
97      defl_yd->add_stl_file( fdefl_yd );
98      geom.set_solid( 12, defl_yd );
99
100     STLSolid *defl_xl = new STLSolid;
101     defl_xl->set_transformation( T1 );
102     defl_xl->add_stl_file( fdefl_xl );
103     geom.set_solid( 13, defl_xl );
104
105     STLSolid *defl_xr = new STLSolid;
106     defl_xr->set_transformation( T1 );
107     defl_xr->add_stl_file( fdefl_xr );
108     geom.set_solid( 14, defl_xr );
109
110     STLSolid *Q13y = new STLSolid;
111     Q13y->set_transformation( T1 );
112     Q13y->add_stl_file( fQ13_y );
113     geom.set_solid( 15, Q13y );
114
115     STLSolid *Q13x = new STLSolid;
116     Q13x->set_transformation( T1 );
117     Q13x->add_stl_file( fQ13_x );
118     geom.set_solid( 16, Q13x );
119
120     STLSolid *Q2y = new STLSolid;
121     Q2y->set_transformation( T1 );
122     Q2y->add_stl_file( fQ2_y );
123     geom.set_solid( 17, Q2y );
124
125     STLSolid *Q2x = new STLSolid;
126     Q2x->set_transformation( T1 );
127     Q2x->add_stl_file( fQ2_x );
128     geom.set_solid( 18, Q2x );
129
130     geom.set_boundary(  1,  Bound(BOUND_NEUMANN,     0.0) ); // x0
131     geom.set_boundary(  2,  Bound(BOUND_NEUMANN,     0.0) ); // x_max
```

```
132     geom.set_boundary(  3,  Bound(BOUND_NEUMANN,     0.0) ); // y_0
133     geom.set_boundary(  4,  Bound(BOUND_NEUMANN,     0.0) ); // y_max
134     geom.set_boundary(  5,  Bound(BOUND_DIRICHLET,   Vex - Vsource) ); // zmin
135     geom.set_boundary(  6,  Bound(BOUND_DIRICHLET,  -Vsource) ); // z_max
136
137     geom.set_boundary(  7,  Bound(BOUND_DIRICHLET,  Vex - Vsource) ); // ex
138
139     geom.set_boundary( 8,  Bound(BOUND_DIRICHLET,  Vbias - Vsource ) ); // el1
140     geom.set_boundary( 9,  Bound(BOUND_DIRICHLET,  Veinzel - Vsource) ); // el2
141     geom.set_boundary( 10,  Bound(BOUND_DIRICHLET,  Vbias - Vsource ) ); // el3
142
143     geom.set_boundary( 11,  Bound(BOUND_DIRICHLET,  +Vdefy/2 - Vsource ) ); // def_y_up
144     geom.set_boundary( 12,  Bound(BOUND_DIRICHLET,  -Vdefy/2 - Vsource ) ); // def_y_down
145     geom.set_boundary( 13,  Bound(BOUND_DIRICHLET,  +Vdefx/2 - Vsource ) ); // def_x_left
146     geom.set_boundary( 14,  Bound(BOUND_DIRICHLET,  -Vdefx/2 - Vsource ) ); // def_x_right
147
148     geom.set_boundary( 15,  Bound(BOUND_DIRICHLET,  +Vq13y - Vsource ) ); // Q triplet 1,3
        vertical
149     geom.set_boundary( 16,  Bound(BOUND_DIRICHLET,  +Vq13x - Vsource ) ); // Q triplet 1,3
        horizontal
150     geom.set_boundary( 17,  Bound(BOUND_DIRICHLET,  +Vq2y - Vsource ) ); // Q triplet 2
        vertical
151     geom.set_boundary( 18,  Bound(BOUND_DIRICHLET,  +Vq2x - Vsource ) ); // Q triplet 2
        horizontal
152
153     geom.build_mesh();
154     geom.save( geom_fn );
155
156 }
157
158
159 int main( int argc, char **argv )
160 {
161     try {
162    //ibsimu.set_message_output( "ibsimu.txt" );
163         ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
164        ibsimu.set_thread_count( 4 );
165        build_geom( argc, argv );
166     } catch( Error e ) {
167        e.print_error_message( ibsimu.message( 0 ) );
168         exit( 1 );
169     }
170
171     return( 0 );
172 }
```

Instead of including the slit aperture in the magnet geometry, a separate section for the slit added to the simulation, which features a finer mesh to accurately simulate the transmission through the small aperture. Because of the 60° magnet deflection, the criterium for particle export to the section is no longer hitting $z_{\max}$, but a grounded "blocker" electrode which acts as a completely closed slit aperture. All particles hitting the "blocker" boundary are transferred to the slit section. In order to preserve z as the particle travel direction, a transformation is necessary after the 60° magnet deflection. We define the $z' = 0$ coordinate

as the magnet pole exit. The total coordinate transformation is therefore $T = T_{\text{rot}} * T_{\text{T}_0,\text{all}} * T_{\text{T}_0,\text{mag}}$, where $T = T_{\text{rot}}$ is the rotation by 60°, $T_{\text{T}_0,\text{all}}$ the z-translation from $z = 0$ to the magnet pipe entrance and $T_{\text{T}_0,\text{mag}}$ the x- and z-translation from magnet pipe entrance to magnet pole exit. T applies to both slit solids (and the virtual blocker). An additional transformation $T_{l,r}$ applies to the left and right segment of the slit, respectively, and opens the slit by *slit_sep* (specified in *slit.sh*).

Listing 7: build_geom_mag.cpp

```
#include <fstream>
#include <iomanip>
#include <limits>
#include <cmath>
#include "dxf_solid.hpp"
#include "stl_solid.hpp"
#include "stlfile.hpp"
#include "mydxffile.hpp"
#include "geometry.hpp"
#include "func_solid.hpp"
#include "error.hpp"
#include "ibsimu.hpp"
#include "meshvectorfield.hpp"

using namespace std;

void build_geom( int argc, char **argv ){

    // get command line arguments
    std::string gfn = argv[1]; // mesh export filename
    std::string stl_path = argv[2]; // stl import path
    const double h = atof(argv[3]); // mesh cell size#
    const double xsize = atof(argv[4]); // full mesh size in x dir in m
    const double ysize = atof(argv[5]); // full mesh size in y dir in m
    const double zmin = atof(argv[6]); //
    const double zmax = atof(argv[7]); // minimum of mesh area in z dir in m (x,y are set automatically)
    const double Vsource = atof(argv[8]); // the magnet does not see the source anymore; so this is set to 0
    const double translate = atof(argv[9]); // extractor(and downstream) elements translation
    const double defl_deg = atof(argv[10]); // deflection angle in deg
    const double slit_dist = atof(argv[11]); // slit distance from magnet (1500 is the dist in drawing)
    //const double slit_sep = atof(argv[12]); // slit separation sd .

    std::string geom_fn = "dat/" + gfn + "/mag_geom_" + gfn + ".dat"; // folder is named for geometry prefix

    double sizereq[3] = { xsize, // x full
                          ysize, // y full
                          zmax - zmin}; // z full
    Int3D meshsize( (int)floor(sizereq[0]/h)+1,
                    (int)floor(sizereq[1]/h)+1,
                    (int)floor(sizereq[2]/h)+1 );
    Vec3D origo( -sizereq[0]+105.0e-3, -sizereq[1]/2, zmin );
```

```
    Geometry geom( MODE_3D, meshsize, origo, h );

    Transformation T1; // applies to Q triplet, magnet
    T1.translate( Vec3D( 0.0, 0.0, translate ) );
    T1.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor export
    settings

    // slits coord transformation
    double defl = M_PI*defl_deg/180.0;
    double z_dist = cos(defl)*(slit_dist);
    double x_dist = sin(defl)*(slit_dist);

    Transformation Tblocker; // blocker
    Tblocker.translate( Vec3D( -x_dist, 0.0, translate + z_dist) );
    Tblocker.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor
    export settings

    STLFile *fmag = new STLFile( stl_path + "sanda_beam_pipe.stl" ); // magnet
    STLFile *fblocker = new STLFile( stl_path + "sanda_blocker.stl" ); // blocker - defines
    particle export plane

    STLSolid *mag = new STLSolid;
    mag->set_transformation( T1 );
    mag->add_stl_file( fmag );
    geom.set_solid( 7, mag );

    STLSolid *blocker = new STLSolid;
    blocker->set_transformation( Tblocker );
    blocker->add_stl_file( fblocker );
    geom.set_solid( 8, blocker );

    geom.set_boundary(  1,  Bound(BOUND_DIRICHLET,     - Vsource) ); // x0
    geom.set_boundary(  2,  Bound(BOUND_DIRICHLET,     - Vsource) ); // x_max
    geom.set_boundary(  3,  Bound(BOUND_DIRICHLET,     - Vsource) ); // y_0
    geom.set_boundary(  4,  Bound(BOUND_DIRICHLET,     - Vsource) ); // y_max
    geom.set_boundary(  5,  Bound(BOUND_DIRICHLET,     - Vsource) ); // zmin
    geom.set_boundary(  6,  Bound(BOUND_DIRICHLET,     - Vsource) ); // z_max

    geom.set_boundary(  7,  Bound(BOUND_DIRICHLET,  - Vsource) ); // mag

    geom.set_boundary( 8,  Bound(BOUND_DIRICHLET,  - Vsource) ); // blocker
    /*
    geom.set_boundary(  9,  Bound(BOUND_DIRICHLET,  - Vsource) ); // slit_l
    geom.set_boundary(  10,  Bound(BOUND_DIRICHLET,  - Vsource) ); // slit_r
    */


    geom.build_mesh();
    geom.save( geom_fn );

}


int main( int argc, char **argv )
{
    try {
```

```
95     //ibsimu.set_message_output( "ibsimu.txt" );
96          ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
97         ibsimu.set_thread_count( 4 );
98         build_geom( argc, argv );
99      } catch( Error e ) {
100        e.print_error_message( ibsimu.message( 0 ) );
101         exit( 1 );
102     }
103
104     return( 0 );
105 }
```

Listing 8: build_geom_sl.cpp

```
1  #include <fstream>
2  #include <iomanip>
3  #include <limits>
4  #include <cmath>
5  #include "dxf_solid.hpp"
6  #include "stl_solid.hpp"
7  #include "stlfile.hpp"
8  #include "mydxffile.hpp"
9  #include "geometry.hpp"
10 #include "func_solid.hpp"
11 #include "error.hpp"
12 #include "ibsimu.hpp"
13 #include "meshvectorfield.hpp"
14
15 using namespace std;
16
17 void build_geom( int argc, char **argv ){
18
19     // get command line arguments //
20     std::string gfn = argv[1]; // mesh export filename
21     std::string stl_path = argv[2]; // stl import path
22     const double h = atof(argv[3]); // mesh cell size
23     const double xsize = atof(argv[4]); // full mesh size in x dir in m
24     const double ysize = atof(argv[5]); // half mesh size in y dir in m
25     const double zmin = atof(argv[6]); //
26     const double zmax = atof(argv[7]); // minimum of mesh area in z dir in m (x,y are set
       automatically)
27     const double Vsource = 0.0; //atof(argv[8]); // the script will shift this to 0 later
       and other voltages accordingly
28     const double mag_zero = atof(argv[9]); // zero coordinate of magnet assembly in mm
29     const double exit_x = atof(argv[10]); // pole (not pipe!) exit coordinates in mm
30     const double exit_z = atof(argv[11]);
31     const double slit_dist = atof(argv[12]); // slit distance from magnet (1500 is the dist
       in drawing)
32     const double slit_sep = atof(argv[13]); // slit separation
33     const double defl_deg = atof(argv[14]); // magnet deflection angle
34
35     std::string geom_fn = "dat/" + gfn + "/sl_geom_" + gfn + ".dat"; // folder is named for
       geometry prefix
36
37     double sizereq[3] = { xsize, // x full
```

```
                        ysize, // y full
                        zmax - zmin}; // z full
    Int3D meshsize( (int)floor(sizereq[0]/h)+1,
                    (int)floor(sizereq[1]/h)+1,
                    (int)floor(sizereq[2]/h)+1 );
    Vec3D origo( -sizereq[0]/2 , -sizereq[1]/2, zmin ); //
    Geometry geom( MODE_3D, meshsize, origo, h );

    // variables
    double defl = M_PI*defl_deg/180.0;

    Transformation T0mag; // from magnet pole exit to magnet pipe entrance
    T0mag.translate( Vec3D( -exit_x, 0.0, -exit_z) );

    Transformation T0all; // from magnet pipe entrace to origin
    T0all.translate( Vec3D( 0.0, 0.0, -mag_zero) );

    Transformation Trot;
    Trot.rotate_y(defl);

    Transformation Tslit = Trot * T0all * T0mag; //

    Transformation Tslit_l = Tslit;
    Tslit_l.translate( Vec3D( -slit_sep/2, 0.0, slit_dist) );
    Tslit_l.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor
    export settings

    Transformation Tslit_r = Tslit;
    Tslit_r.translate( Vec3D( +slit_sep/2, 0.0, slit_dist) );
    Tslit_r.scale( Vec3D( 1.0e-3, 1.0e-3, 1.0e-3 ) ); // corresponds to "m" in inventor
    export settings

    STLFile *fslits_l = new STLFile( stl_path + "sanda_slits_l.stl" ); // left slit
    STLFile *fslits_r = new STLFile( stl_path + "sanda_slits_r.stl" ); // right slit

    STLSolid *slits_l = new STLSolid;
    slits_l->set_transformation( Tslit_l );
    slits_l->add_stl_file( fslits_l );
    geom.set_solid( 7, slits_l );

    STLSolid *slits_r = new STLSolid;
    slits_r->set_transformation( Tslit_r );
    slits_r->add_stl_file( fslits_r );
    geom.set_solid( 8, slits_r );

    geom.set_boundary(  1,  Bound(BOUND_NEUMANN,     - Vsource) ); // x0
    geom.set_boundary(  2,  Bound(BOUND_NEUMANN,     - Vsource) ); // x_max
    geom.set_boundary(  3,  Bound(BOUND_NEUMANN,     - Vsource) ); // y_0
    geom.set_boundary(  4,  Bound(BOUND_NEUMANN,     - Vsource) ); // y_max
    geom.set_boundary(  5,  Bound(BOUND_NEUMANN,     - Vsource) ); // zmin
    geom.set_boundary(  6,  Bound(BOUND_NEUMANN,     - Vsource) ); // z_max

    geom.set_boundary(  7,  Bound(BOUND_DIRICHLET,  - Vsource) ); // slit_l
    geom.set_boundary(  8,  Bound(BOUND_DIRICHLET,  - Vsource) ); // slit_r

    geom.build_mesh();
```

```
92      geom.save( geom_fn );
93
94  }
95
96
97  int main( int argc, char **argv )
98  {
99      try {
100    //ibsimu.set_message_output( "ibsimu.txt" );
101         ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
102        ibsimu.set_thread_count( 4 );
103        build_geom( argc, argv );
104     } catch( Error e ) {
105        e.print_error_message( ibsimu.message( 0 ) );
106         exit( 1 );
107     }
108
109     return( 0 );
110 }
```

The *plot_geom.cpp* utility can be used to plot the simulation mesh of each section (by changing the first command line argument in *master.sh*, line 123). An interactive viewer is opened on successful execution. Note that this requires a functional XServer (such as XMing). For details refer to the included IBSimu installation guide.

Listing 9: plot_geom.cpp

```
1   #include <fstream>
2   #include <iomanip>
3   #include <limits>
4   #include "meshvectorfield.hpp"
5   #include "dxf_solid.hpp"
6   #include "mydxffile.hpp"
7   #include "gtkplotter.hpp"
8   #include "geomplotter.hpp"
9   #include "geometry.hpp"
10  #include "func_solid.hpp"
11  #include "epot_efield.hpp"
12  #include "error.hpp"
13  #include "ibsimu.hpp"
14  #include "trajectorydiagnostics.hpp"
15  #include "particledatabase.hpp"
16  #include "particlediagplotter.hpp"
17
18
19
20  using namespace std;
21
22  void simu( int argc, char **argv )
23  {
24      std::string prefix = argv[1];
25      std::string gfn = argv[2];
26      std::string geom_fn = "dat/" + gfn + "/" + prefix + "_geom_" + gfn + ".dat";//
27
28      std::ifstream is_geom( geom_fn );
```

```
    if( !is_geom.good() )
   throw( Error( ERROR_LOCATION, (string)"couldn\'t open file \'" + geom_fn + "\'" ) );
    Geometry geom( is_geom );
    is_geom.close();
    geom.build_surface();
//
    GTKPlotter plotter( &argc, &argv );
    plotter.set_geometry( &geom );
    plotter.new_geometry_plot_window();
    plotter.run();
}


int main( int argc, char **argv )
{
    if( argc < 1 ) {
   cerr << "Usage: analysis geom\n";
   exit( 1 );
    }

    try {
   ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
   ibsimu.set_thread_count( 4 );
   simu( argc, argv );
    } catch( Error e ) {
   e.print_error_message( ibsimu.message( 0 ) );
        exit( 1 );
    }

    return( 0 );
}
```

The *simu_<section>.cpp* files contain the actual simulation of electric fields and particle trajectories. Particle collisions to the boundaries defined in *build_geom_<section>.cpp* are reported in the log-file and terminal output. Particles arriving at boundary 6 ($= z_{\max}$) are exported for transfer to the next simulation section, or plotting. In the case of *simu_ex.cpp* the particle database is initially generated, with articles starting at a random position within the atomizer volume with a velocities according the a Maxwellian distribution. In order to simulate the electric field within the atomizer, the IBSimu builtin plasma model is used. Plasma particles are referred to as "surface ions". For a detailed discussion see Sec. 3.3.1.

Listing 10: simu_ex.cpp

```
#include <fstream>
#include <cstdio>
#include <iomanip>
#include <limits>
#include <string>
#include <random>
#include <cmath>
#include <vector>
#include "epot_bicgstabsolver.hpp"
#include "meshvectorfield.hpp"
#include "dxf_solid.hpp"
```

```
12 #include "stl_solid.hpp"
13 #include "stlfile.hpp"
14 #include "mydxffile.hpp"
15 #include "geometry.hpp"
16 #include "func_solid.hpp"
17 #include "epot_efield.hpp"
18 #include "particledatabase.hpp"
19 #include "error.hpp"
20 #include "ibsimu.hpp"
21 #include "config.h"
22 #include "plasma.h"
23
24 using namespace std;
25
26 class SlotLine {
27     public:
28         double r; // radius
29         double h; // height
30         double l; // length (z)
31         //double eps;
32
33         void set_r (double radius) { r = radius; }
34         void set_h (double height) { h = height; }
35         void set_l (double length) { l = length; }
36         //void set_eps (double epsilon) { eps = epsilon; }
37         double get_r (void) { return r; }
38         double get_h (void) { return h; }
39         double get_l (void) { return l; }
40         //double get_eps (void) { return eps; }
41
42         bool inside (double x, double y, double z, double eps){
43         // returns true if given tuple of coordinates (x,y,z) lies within line volume
44         // with 'eps' distance to walls/exit
45         return (    ( x*x + (y-h/2)*(y-h/2) < (r-eps)*(r-eps) ) || // upper circle (x,y)
46                     ( y < h/2-eps && abs(x) < r-eps )   )   // slot (x,y)
47                 &&  ( z < 0.0-eps && z > -l+eps );  // z
48         }
49 };
50
51
52 void simu( int argc, char **argv )
53 {
54     std::string gfn = argv[1];
55     std::string sfn = argv[2];
56     const double r_line = atof(argv[3]); // line inner radius
57     const double h_line = atof(argv[4]); // line height (consider a slot hole with length
       h_line and radius r_line; set 0 for circular line)
58     const double l_line = atof(argv[5]); // ion beam z start position
59     const double eps0 = atof(argv[6]); // distance of LASER ION particle start position
       from line wall
60     const double V_line = M_PI*r_line*r_line*l_line + 2*r_line*h_line*l_line; // line volume
61     //const double V_line_part = M_PI*(r_line-eps)*(r_line-eps)*l_line +
       2*(r_line-eps)*(h_line-eps)*l_line; // particle start volume
62     const double zmax = atof(argv[7]); // max. z coord; needed for particle export
63     const bool use_ibsimu_plasma = atoi(argv[8]); // use ibsimu pexp plasma model (1) or
       not (0)
```

```
64      const bool use_analytic_plasma = atoi(argv[9]); // use analytic plasma model (1) not (0)
65      const bool show_surface_ions = atoi(argv[10]); // whether to generate tracetories for
        surface ions or fix them stationary
66      const double z_plasma = atof(argv[11]); // plasma at z < zplasma
67      const double W = atof(argv[12]); // Atomizer Material work function in eV
68      const double Vplasma = atof(argv[13]); //Plasma potential, from Kirchner 1990 (Tantalum)
69      const double Te = atof(argv[14]); // electron temperature (in K)
70      const double Ti = atof(argv[15]); // surface ion temperature (in K)
71      const double T0 = atof(argv[16]); // laser ion temperature (in K)
72      const double U_line = atof(argv[17]); // line voltage for additional velocity in z
        direction
73      const double Ni = atof(argv[18]); // number of surface ion macro particles
74      const double N0 = atof(argv[19]); // number of laser ion macro particles
75      const double massi = atof(argv[20]); // microscopic surface ion particle mass in u
76      const double mass0 = atof(argv[21]); // microscopic laser ion particle mass in u
77      const double ni = atof(argv[22]); // bulk plasma (surface) ion density in cm^-3
78      const double I0 = atof(argv[23]); // unit of total laser ion current
79      const double iterations = atoi(argv[24]); // number of Vlasov iterations
80      const double tol = atof(argv[25]); // electric field solver convergence tolerance
81      const double sc_alpha = atof(argv[26]); // space charge averaging parameter
82
83      // print args
84      for (int i = 0; i < argc; i++){
85          std::string arg_i = argv[i];
86          cout << "argv " << i << " = " << arg_i << "\n";
87      }
88
89      // import/export filenames
90      std::string geom_fn = "dat/" + gfn + "/ex_geom_" + gfn + ".dat";
91      std::string epot_fn = "dat/" + gfn + "/ex_epot_" + sfn + ".dat";
92      std::string pdb_fn = "dat/" + gfn + "/ex_pdb_" + sfn + ".dat";
93      std::string scharge_fn = "dat/" + gfn + "/ex_scharge_" + sfn + ".dat";
94      std::string particles_out_fn = "dat/" + gfn + "/ex_particles_" + sfn + ".txt";
95
96      /* read geom.dat */
97      std::ifstream is_geom( geom_fn.c_str() );
98      if( !is_geom.good() )//
99          throw( Error( ERROR_LOCATION, (std::string)"couldn\'t open file \'" + geom_fn +
        "\'" ) );
100     Geometry geom( is_geom );
101     is_geom.close();
102     geom.build_surface();
103
104     // initialize Line class
105     SlotLine Line;
106     Line.set_r(r_line);
107     Line.set_h(h_line);
108     Line.set_l(l_line);
109
110     // init solver
111     EpotBiCGSTABSolver solver( geom, tol, 10000, 1e-4, 100 ); // solver object (type
        depends on geometry)
112     InitialPlasma initp( AXIS_Z, z_plasma ); // initial plasma at z < zPlasma
113     if (use_ibsimu_plasma){
114         solver.set_initial_plasma( Vplasma, &initp ); // consider different functions in
        EpotBiCGSTABSolver Class Reference
```

```
115     }
116     solver.set_gnewton( true ); // Enable/disable globally convergent Newton-Raphson - true
        is default
117
118     // init efield,bfield and space charge
119     EpotField epot( geom );
120     MeshVectorField bfield;
121     MeshScalarField scharge( geom );
122     MeshScalarField scharge_ave( geom ); // for space charge averaging (see radis H-)
123     EpotEfield efield( epot );
124     field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
125                 FIELD_SYMMETRIC_POTENTIAL, FIELD_EXTRAPOLATE,
126                 FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
127     efield.set_extrapolation( efldextrpl );
128
129     // init particle database
130     ParticleDataBase3D pdb( geom );
131     pdb.set_max_steps( 1000 );
132     bool pmirror[6] = { false, false, true, false, false, false };
133     pdb.set_mirror( pmirror );
134     pdb.set_surface_collision( true ); // not in original "Radis", needed or prgram crashes
        when loading a pre-built geom
135
136     // Vlasov iterations
137     for( size_t i = 0; i < iterations; i++ ) { //originally i<15
138          ibsimu.message(1) << "\n --------------------------- " + sfn + " --- Extraction
        --- Vlasov loop i = " << i << "\n"; // loop count output
139
140         // Space charge averaging & Plasma
141        if( i == 1 && use_ibsimu_plasma ) { // 1 is correct !! see Plasmacyl/RadisH-
142             double rhoe = pdb.get_rhosum();
143             //double rhoe = richardson(W, Te) * 1.602e-19; // electron surface density in
        C/m^3
144             cout << "#####################################################\n Plasma Ion
        Rhosum = " << pdb.get_rhosum()
145                 << "\n#####################################################\n";
146             cout << "#####################################################\n Electron Charge
        density = " << rhoe
147                 << "\n#####################################################\n";
148             double Te_eV = Te  * 0.00008617328149741; // to eV
149             solver.set_pexp_plasma( rhoe, Te_eV, Vplasma ); // taken from Plasmacyl
150        }
151
152
153         solver.solve( epot, scharge_ave ); // when SC averaging is removed, put scharge here
154         // add calculated plasma potential to Epot
155         if (use_analytic_plasma){
156             for( uint32_t k = 0; k < geom.size(0); k++ ) {
157               for( uint32_t l = 0; l < geom.size(1); l++ ) {
158                     for( uint32_t m = 0; m < geom.size(2); m++ ) {
159                         double xx = geom.origo(0) + geom.h()*k;
160                         double yy = geom.origo(1) + geom.h()*l;
161                         double zz = geom.origo(2) + geom.h()*m;
162                         if (zz < z_plasma){
163                             epot(k,l,m) += phi_cav(xx, yy, W, Ti, 1.0e6*ni, 2*r_line);
164                         }
```

```
                }
            }
        }
    }
    //solve poisson
   efield.recalculate();
    pdb.clear();


    // random number stuff
    std::random_device rd;  //Will be used to obtain a seed for the random number engine
    std::mt19937 gen(rd()); //Standard mersenne_twister_engine seeded with rd()
    // SURFACE IONS
    // derived parameters from beam
    double sigma_i = sqrt( 1.38e-23 * Ti / (massi * 1.66e-27) ); // gaussian velocity distribution stddev
    double epsi = 0.0e-3; // approx 1 mesh cell?
    std::uniform_real_distribution<double> unif_x_i(-r_line+epsi, r_line-epsi); // random distribution X (lower, upper)
    std::uniform_real_distribution<double> unif_y_i(0.0, h_line/2 + r_line -epsi); // random distribution Y (lower, upper)
    std::uniform_real_distribution<double> unif_z_i(-l_line+epsi, 0.0-epsi); // random distribution Z (lower, upper)
    std::normal_distribution<double> gauss_v_i(0.0, sigma_i); // gaussian distribution for velocity components (center 0, stddev sigma)
    // LASER IONS
    double sigma_0 = sqrt( 1.38e-23 * T0 / (mass0 * 1.66e-27) ); // gaussian velocity distribution stddev
    std::uniform_real_distribution<double> unif_x_0(-r_line+eps0, r_line-eps0); // random distribution X (lower, upper)
    std::uniform_real_distribution<double> unif_y_0(0.0, h_line/2 + r_line -eps0); // random distribution Y (lower, upper)
    std::uniform_real_distribution<double> unif_z_0(-l_line+eps0, 0.0-eps0); // random distribution Z (lower, upper)
    std::normal_distribution<double> gauss_v_0(0.0, sigma_0); // gaussian distribution for velocity components (center 0, stddev sigma)
    std::discrete_distribution<int> dmass0 {1,1,0,1}; // for mass, weights for (m-1, m, m+1) in brackets

    // ##################### SURFACE IONS ##############################
    int j = 0; // loop variable
    do {
        if (ni == 0) {break;} // dont add particles
        if (Ni == 0) {break;} // dont add particles

        // conversions
        double ni_m3 = 1.0e6 * ni; // convert density from cm-3 to m-3
        double rho_i = 1.602e-19 * ni_m3; // convert (e)/m^3 to C/m^3
        double Ii = rho_i * 0.5* V_line; // charge of plasma ions within line volume (time independent simulation charge=current)

        // rando, coordinates
        double x_coord = unif_x_i(gen); // random x coord
        double y_coord = unif_y_i(gen); // random y coord
        double z_coord = unif_z_i(gen); // random y coord
        double mass_i = massi;
```

```
209
210         // check if coordinate is valid (x,y only, z should always be ok) --> create
    particle, otherwise not increase j
211         if( Line.inside(x_coord, y_coord, z_coord, epsi) ){
212             // cout << " adding particle " << j << "\r"; // loop count output
213             pdb.add_particle(Ii/(Ni),   // current
214                              1,       // charge state of microscopic particle
215                              mass_i,   // mass in u
216                              ParticleP3D(0, // time (??)
217                                          x_coord, // x
218                                          gauss_v_i(gen), // vx, random gaussian distr.
    velocity
219                                          y_coord, // y
220                                          gauss_v_i(gen), // vy
221                                          z_coord, // z
222                                          gauss_v_i(gen) /*+ vz*/ )); // vz with
    additional component from line voltage
223             j++;
224             pdb.set_rhosum(pdb.get_rhosum() // old rhosum in C (?)
225             + Ii/(Ni) );  // plus charge of macro particle
226         }
227     } while (j < Ni);
228
229     // set coll for surface ions to render them stationary; will establish plasma
    without particle trajectories
230     if(!show_surface_ions){
231         for( size_t k = 0; k < pdb.size(); k++ ) {
232             Particle3D &pp = pdb.particle( k );
233             pp.set_status(PARTICLE_COLL);
234         }
235     }
236
237     // ######################## LASER IONS ############################
238     j = 0; // loop variable
239     do {
240         if (I0 == 0) {break;}
241         if (N0 == 0) {break;} // dont add particles
242
243         // rando, coordinates
244         double x_coord = unif_x_0(gen); // random x coord
245         double y_coord = unif_y_0(gen); // random y coord
246         double z_coord = unif_z_0(gen); // random y coord
247         double mass_0 = (mass0-1.0) + (double)dmass0(gen); // radom mass with m=mass pm
    1
248
249         double U_line_eff = U_line * abs(z_coord)/l_line; // effective acceleration
    voltage within line volume depends on ion creation z coord
250         double vz = sqrt( 2 * 1.6e-19 /*e*/ * U_line_eff / (mass_0 * MASS_U /*amu*/ )
    ); // additional beam energy in z direction; from kinetic energy of E = eU = 0.5mv^2
251
252         // check if coordinate is valid (x,y only, z should always be ok) --> create
    particle, otherwise not increase j
253         if( Line.inside(x_coord, y_coord, z_coord, eps0) ){
254             // cout << " adding particle " << j << "\r"; // loop count output
255             pdb.add_particle(I0/(2*N0),   // current (2*N because of mirror symmetry)
256                              1,       // charge state of microscopic particle
```

```
                                mass_0,   // mass in u
                                ParticleP3D(0, // time (??)
                                            x_coord, // x
                                            gauss_v_0(gen), // vx, random gaussian distr. velocity
                                            y_coord, // y
                                            gauss_v_0(gen), // vy
                                            z_coord, // z
                                            gauss_v_0(gen) + vz )); // vz with additional component from line voltage
                    j++;
                    // LASER IONS ARE NOT ADDED TO RHOSUM TO AVOID COMPENSATINMG ELECTRON CHARGE IN PLASMA
                }
            } while (j < N0);

            pdb.iterate_trajectories( scharge, efield, bfield ); // error "solid 5 not defined" occurs here when pdb.set_surface_collision( false );

            // Space charge averaging (RADIS H-)
           if( i == 0 ) {
                scharge_ave = scharge;
           } else {
                double sc_beta = 1.0-sc_alpha;
                uint32_t nodecount = scharge.nodecount();
                for( uint32_t b = 0; b < nodecount; b++ ) {
                   scharge_ave(b) = sc_alpha*scharge(b) + sc_beta*scharge_ave(b);
                }
           }
            //pdb.reset_trajectories();
        }

        // export
        epot.save( epot_fn );
        pdb.save( pdb_fn );
        scharge_ave.save( scharge_fn );

        // Write output file containing all particles
        ofstream fileOut( particles_out_fn );
        for( size_t k = 0; k < pdb.size(); k++ ) {

            Particle3D &pp = pdb.particle( k );
            // Skip electrons
            if( pp.m() < 0.5*MASS_U )
                continue;
            // Skip ions not at the end
            if( pp(PARTICLE_Z) < (zmax - 1e-3 ) ) // 1 mm tolerance (at least mesh size h needed as tolerance)
                continue;

            // Plot particle I, m, coordinates##
            // 3D has 7 coordinates
            fileOut << setw(12) << pp.IQ() << " ";
            fileOut << setw(12) << pp.m() << " ";
            for( size_t j = 0; j < 7; j ++ )
                fileOut << setw(12) << pp(j) << " ";
```

```
308         fileOut << "\n";
309     }
310     fileOut.close();
311 
312 }
313 
314 int main( int argc, char **argv )
315 {
316     try {
317         ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
318        ibsimu.set_thread_count( 4 );
319        simu( argc, argv );
320     } catch( Error e ) {
321        e.print_error_message( ibsimu.message( 0 ) );
322         exit( 1 );
323     }
324 
325     return( 0 );
326 }
```

Listing 11: simu_io.cpp

```
1  #include <fstream>
2  #include <cstdio>
3  #include <iomanip>
4  #include <limits>
5  #include <string>
6  #include <random>
7  #include <cmath>
8  #include <vector>
9  #include <readascii.hpp>
10 #include "epot_bicgstabsolver.hpp"
11 #include "meshvectorfield.hpp"
12 #include "dxf_solid.hpp"
13 #include "stl_solid.hpp"
14 #include "stlfile.hpp"
15 #include "mydxffile.hpp"
16 #include "geometry.hpp"
17 #include "func_solid.hpp"
18 #include "epot_efield.hpp"
19 #include "particledatabase.hpp"
20 #include "error.hpp"
21 #include "ibsimu.hpp"
22 #include "config.h"
23 
24 using namespace std;
25 
26 void simu( int argc, char **argv )
27 {
28     std::string gfn = argv[1];
29     std::string sfn = argv[2];
30     const double zmax = atof(argv[3]); // max. z coord; needed for particle export ##
31     const double iterations = atoi(argv[4]); // number of Vlasov iterations
32     const double tol = atof(argv[5]); // electric field solver convergence tolerance
33     const double sc_alpha = atof(argv[6]); // space charge averaging parameter
```

```
34
35      // import/export filenames
36      std::string geom_fn = "dat/" + gfn + "/io_geom_" + gfn + ".dat";
37      std::string epot_fn = "dat/" + gfn + "/io_epot_" + sfn + ".dat";
38      std::string pdb_fn = "dat/" + gfn + "/io_pdb_" + sfn + ".dat";
39      std::string scharge_fn = "dat/" + gfn + "/io_scharge_" + sfn + ".dat";
40      std::string particles_in_fn = "dat/" + gfn + "/ex_particles_" + sfn + ".txt";
41      std::string particles_out_fn = "dat/" + gfn + "/io_particles_" + sfn + ".txt";
42
43      /* read geom.dat */
44      std::ifstream is_geom( geom_fn.c_str() );
45      if( !is_geom.good() )//
46          throw( Error( ERROR_LOCATION, (std::string)"couldn\'t open file \'" + geom_fn +
        "\'" ) );
47      Geometry geom( is_geom );
48      is_geom.close();
49      geom.build_surface();
50
51      // init solver
52      EpotBiCGSTABSolver solver( geom, tol ); // solver object (type depends on geometry)
53      solver.set_gnewton( true ); // Enable/disable globally convergent Newton-Raphson (does
        nothing ???) - true is default
54
55      // init efield,bfield and space charge
56      EpotField epot( geom );
57      MeshVectorField bfield;
58      MeshScalarField scharge( geom );
59      MeshScalarField scharge_ave( geom ); // for space charge averaging (see radis H-)
60      EpotEfield efield( epot );
61      field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
62                  FIELD_SYMMETRIC_POTENTIAL, FIELD_EXTRAPOLATE,
63                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
64      efield.set_extrapolation( efldextrpl );
65
66      // init particle database
67      ParticleDataBase3D pdb( geom );
68      pdb.set_max_steps( 1000 );
69      bool pmirror[6] = { false, false, true, false, false, false };
70      pdb.set_mirror( pmirror );
71      pdb.set_surface_collision( true ); // not in original "Radis", needed or prgram crashes
        when loading a pre-built geom
72
73      // Vlasov iterations
74      for( size_t i = 0; i < iterations; i++ ) { //originally i<15
75           ibsimu.message(1) << "\n --------------------------- " + sfn + " --- Ion optics
        --- Vlasov loop i = " << i << "\n"; // loop count output
76
77          //solve poisson
78         solver.solve( epot, scharge_ave ); // when SC averaging is removed, put scharge here
79         efield.recalculate();
80          pdb.clear();
81
82          // Input particles
83          ReadAscii din( particles_in_fn, 9 );
84          //cout << "Reading " << din.rows() << " particles\n";
85          for( size_t i = 0; i < din.rows(); i++ ) {
```

```
            double I  = din[0][i];
            double m  = din[1][i];
            double t  = din[2][i];
            double x  = din[3][i];
            double vx = din[4][i];
            double y  = din[5][i];
            double vy = din[6][i];
            double z  = din[7][i];
            double vz = din[8][i];

            pdb.add_particle( I, 1.0, m/MASS_U, ParticleP3D(t,x,vx,y,vy,z,vz) );
        }


        pdb.iterate_trajectories( scharge, efield, bfield ); // error "solid 5 not defined" occurs here when pdb.set_surface_collision( false );

        // Space charge averaging (RADIS H-)
       if( i == 0 ) {
           scharge_ave = scharge;
       } else {
            uint32_t nodecount = scharge.nodecount();
            for( uint32_t b = 0; b < nodecount; b++ ) {
                scharge_ave(b) = sc_alpha*scharge(b) + (1.0-sc_alpha)*scharge_ave(b);
            }
       }
    }

    // export
    epot.save( epot_fn );
    pdb.save( pdb_fn );
    scharge_ave.save( scharge_fn );

    // Write output file containing all particles
    ofstream fileOut( particles_out_fn );
    for( size_t k = 0; k < pdb.size(); k++ ) {

        Particle3D &pp = pdb.particle( k );
        // Skip electrons
        if( pp.m() < 0.5*MASS_U )
            continue;
        // Skip ions not at the end
        if( pp(PARTICLE_Z) < (zmax- 3e-3) ) // 3 mm tolerance (at least mesh size h needed as tolerance)
            continue;

        // Plot particle I, m, coordinates
        // 3D has 7 coordinates
        fileOut << setw(12) << pp.IQ() << " ";
        fileOut << setw(12) << pp.m() << " ";
        for( size_t j = 0; j < 7; j ++ )
            fileOut << setw(12) << pp(j) << " ";
        fileOut << "\n";
    }
    fileOut.close();
}
```

```
140
141 int main( int argc, char **argv )
142 {
143     try {
144         ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
145        ibsimu.set_thread_count( 4 );
146        simu( argc, argv );
147     } catch( Error e ) {
148        e.print_error_message( ibsimu.message( 0 ) );
149         exit( 1 );
150     }
151
152     return( 0 );
153 }
```

simu_mag.cpp includes the import of magnetic field data. The data should be provided in the from (x, y, z, B_x, B_y, B_z), where coordinates are interpreted as millimeters and field value as Tesla. The field values can be linearly scaled by adjusting the *bscale* variable. The *mag_zero* variable translates the magnetic field origin along the z-directions. Other transformations, such as rotations, are not applied, but can in principle be implemented.

Listing 12: simu_mag.cpp

```
1  #include <fstream>
2  #include <cstdio>
3  #include <iomanip>
4  #include <limits>
5  #include <string>
6  #include <random>
7  #include <cmath>
8  #include <vector>
9  #include <readascii.hpp>
10 #include "epot_bicgstabsolver.hpp"
11 #include "meshvectorfield.hpp"
12 #include "dxf_solid.hpp"
13 #include "stl_solid.hpp"
14 #include "stlfile.hpp"
15 #include "mydxffile.hpp"
16 #include "geometry.hpp"
17 #include "func_solid.hpp"
18 #include "epot_efield.hpp"
19 #include "particledatabase.hpp"
20 #include "error.hpp"
21 #include "ibsimu.hpp"
22 #include "config.h"
23
24 using namespace std;
25
26 void simu( int argc, char **argv )
27 {
28     std::string gfn = argv[1];
29     std::string sfn = argv[2];
30     std::string bfn = argv[3];
31     const double bscale = atof(argv[4]);
```

```
32      const double xdiag = atof(argv[5]); // for export of particles ##
33      const double mag_zero = atof(argv[6]); // zero Z of magnet assembly
34      const double iterations = atoi(argv[7]); // number of Vlasov iterations
35      const double tol = atof(argv[8]); // electric field solver convergence tolerance
36      const double sc_alpha = atof(argv[9]); // space charge averaging parameter
37
38
39      // import/export filenames
40      std::string geom_fn = "dat/" + gfn + "/mag_geom_" + gfn + ".dat";
41      std::string epot_fn = "dat/" + gfn + "/mag_epot_" + sfn + ".dat";
42      std::string pdb_fn = "dat/" + gfn + "/mag_pdb_" + sfn + ".dat";
43      std::string bfieldfn = bfn;
44      std::string scharge_fn = "dat/" + gfn + "/mag_scharge_" + sfn + ".dat";
45      std::string particles_in_fn = "dat/" + gfn + "/io_particles_" + sfn + ".txt";
46      std::string particles_out_fn = "dat/" + gfn + "/mag_particles_" + sfn + ".txt";
47
48      /* read geom.dat */
49      std::ifstream is_geom( geom_fn.c_str() );
50      if( !is_geom.good() )//
51          throw( Error( ERROR_LOCATION, (std::string)"couldn\'t open file \'" + geom_fn +
        "\'" ) );
52      Geometry geom( is_geom );
53      is_geom.close();
54      geom.build_surface();
55
56      // init solver
57      EpotBiCGSTABSolver solver( geom, tol ); // solver object (type depends on geometry)
58      solver.set_gnewton( true ); // Enable/disable globally convergent Newton-Raphson (does
        nothing ???) - true is default
59
60      // init efield and space charge
61      EpotField epot( geom );
62      MeshScalarField scharge( geom );
63      MeshScalarField scharge_ave( geom ); // for space charge averaging (see radis H-)
64      EpotEfield efield( epot );
65      field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
66                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE, // FIELD_SYMMETRIC_POTENTIAL
67                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
68      efield.set_extrapolation( efldextrpl );
69
70      // Define magnetic field
71      bool fout[3] = {true, true, true};
72      //MeshVectorField bfield;
73      MeshVectorField bfield( MODE_3D, fout, 1.0e-3, bscale, bfieldfn ); // mode, which
        coords should be read[3], coord scale, field scale, filename (with lines x,y,z,bx,by,bz)
74      /* RISIKO */
75      bfield.translate( Vec3D(0.0, 0.0e-3, 1e-3*mag_zero )); // ALSO ADJUST IN ANALYSIS.CPP
76      field_extrpl_e bfldextrpl[6] = { FIELD_ZERO, FIELD_ZERO,
77                                       FIELD_ZERO, FIELD_ZERO,
78                                       FIELD_ZERO, FIELD_ZERO };
79      bfield.set_extrapolation( bfldextrpl );
80
81      // init particle database
82      ParticleDataBase3D pdb( geom );
83      pdb.set_max_steps( 1000 );
84      bool pmirror[6] = { false, false, false, false, false, false };
```

```
85      pdb.set_mirror( pmirror );
86      pdb.set_surface_collision( true ); // not in original "Radis", needed or prgram crashes
        when loading a pre-built geom
87
88      // Vlasov iterations
89      for( size_t i = 0; i < iterations; i++ ) { //originally i<15
90           ibsimu.message(1) << "\n ---------------------------- " + sfn + " --- Magnet ---
        Vlasov loop i = " << i << "\n"; // loop count output
91
92          //solve poisson
93         solver.solve( epot, scharge_ave ); // when SC averaging is removed, put scharge here
94         efield.recalculate();
95          pdb.clear();
96
97          // Input particles
98          ReadAscii din( particles_in_fn, 9 );
99          //cout << "Reading " << din.rows() << " particles\n";
100         for( size_t i = 0; i < din.rows(); i++ ) {
101             double I  = din[0][i];
102             double m  = din[1][i];
103             double t  = din[2][i];
104             double x  = din[3][i];
105             double vx = din[4][i];
106             double y  = din[5][i];
107             double vy = din[6][i];
108             double z  = din[7][i];
109             double vz = din[8][i];
110
111             pdb.add_particle( I, 1.0, m/MASS_U, ParticleP3D(t,x,vx,y,vy,z,vz) );
112             pdb.add_particle( I, 1.0, m/MASS_U, ParticleP3D(t,x,vx,-y,-vy,z,vz) );
113         }
114
115
116         pdb.iterate_trajectories( scharge, efield, bfield ); // error "solid 5 not defined"
        occurs here when pdb.set_surface_collision( false );
117
118         // Space charge averaging (RADIS H-)
119        if( i == 0 ) {
120            scharge_ave = scharge;
121        } else {
122             uint32_t nodecount = scharge.nodecount();
123             for( uint32_t b = 0; b < nodecount; b++ ) {
124                 scharge_ave(b) = sc_alpha*scharge(b) + (1.0-sc_alpha)*scharge_ave(b);
125             }
126        }
127     }
128
129     // export
130     epot.save( epot_fn );
131     pdb.save( pdb_fn );
132     scharge_ave.save( scharge_fn );
133
134     // Write output file containing all particles
135     ofstream fileOut( particles_out_fn );
136     for( size_t k = 0; k < pdb.size(); k++ ) {
137
```

```
        Particle3D &pp = pdb.particle( k );
        // Skip electrons
        if( pp.m() < 0.5*MASS_U )
            continue;
        // Skip ions that did not hit the blocker (solid #14)
        Vec3D loc = pp.location();
        //if( pp(PARTICLE_X) > xdiag )
        //    continue;

        // Plot particle I, m, coordinates
        // 3D has 7 coordinates
        fileOut << setw(12) << pp.IQ() << " ";
        fileOut << setw(12) << pp.m() << " ";
        for( size_t j = 0; j < 7; j ++ )
            fileOut << setw(12) << pp(j) << " ";
        fileOut << "\n";
    }
    fileOut.close();
}

int main( int argc, char **argv )
{
    try {
        ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
       ibsimu.set_thread_count( 4 );
       simu( argc, argv );
    } catch( Error e ) {
       e.print_error_message( ibsimu.message( 0 ) );
        exit( 1 );
    }

    return( 0 );
}
```

Particles imported to *simu_sl* are transformed to preserve travel in z-direction. For details see explanation on listing 7: *build_geom_mag.cpp*.

Listing 13: simu_sl.cpp

```
#include <fstream>
#include <cstdio>
#include <iomanip>
#include <limits>
#include <string>
#include <random>
#include <cmath>
#include <vector>
#include <readascii.hpp>
#include "epot_bicgstabsolver.hpp"
#include "meshvectorfield.hpp"
#include "dxf_solid.hpp"
#include "stl_solid.hpp"
#include "stlfile.hpp"
#include "mydxffile.hpp"
#include "geometry.hpp"
```

```
17 #include "func_solid.hpp"
18 #include "epot_efield.hpp"
19 #include "particledatabase.hpp"
20 #include "error.hpp"
21 #include "ibsimu.hpp"
22 #include "config.h"
23 #include <trajectorydiagnostics.hpp>
24
25 using namespace std;
26
27 void simu( int argc, char **argv )
28 {
29     std::string gfn = argv[1];
30     std::string sfn = argv[2];
31     double mag_zero = atof(argv[3]); // zero coordinate of magnet assembly in mm ##
32     double exit_x = atof(argv[4]); // pole (not pipe!) exit coordinates in mm
33     double exit_z = atof(argv[5]);
34     double slit_dist = atof(argv[6]); // slit distance from magnet (1500 is the dist in
       drawing)
35     //const double slit_sep = atof(argv[7]); // slit separation
36     const double defl_deg = atof(argv[8]); // magnet deflection angle
37     //const double zdiag = atof(argv[9]); // for export of particles
38     const double filter = atof(argv[10]); //
39     const double iterations = atoi(argv[11]); // number of Vlasov iterations
40     const double tol = atof(argv[12]); // electric field solver convergence tolerance
41     const double sc_alpha = atof(argv[13]); // space charge averaging parameter
42
43     // import/export filenames
44     std::string geom_fn = "dat/" + gfn + "/sl_geom_" + gfn + ".dat";
45     std::string epot_fn = "dat/" + gfn + "/sl_epot_" + sfn + ".dat";
46     std::string pdb_fn = "dat/" + gfn + "/sl_pdb_" + sfn + ".dat";
47     std::string scharge_fn = "dat/" + gfn + "/sl_scharge_" + sfn + ".dat";
48     std::string particles_in_fn = "dat/" + gfn + "/mag_particles_" + sfn + ".txt";
49     std::string particles_out_fn = "dat/" + gfn + "/sl_particles_" + sfn + ".txt";
50     std::string particles_out_ALL_fn = "dat/" + gfn + "/sl_particles_ALL_" + sfn + ".txt";
51
52     // scale
53     slit_dist *= 1e-3;
54     exit_x *= 1e-3;
55     exit_z *= 1e-3;
56     mag_zero *= 1e-3;
57
58     // Transformation for particle inmput
59     double defl = M_PI*defl_deg/180.0;
60     double z_dist = cos(defl)*slit_dist;
61     double x_dist = sin(defl)*slit_dist;
62
63     Transformation Tblocker1; // inverse of Tblocker in build_geom_mag (only needed for
       particles)
64     Tblocker1.translate( Vec3D( x_dist, 0.0, -z_dist) );
65
66     Transformation T0mag; // from magnet pole exit to magnet pipe entrance
67     T0mag.translate( Vec3D( -exit_x, 0.0, -exit_z) );
68
69     Transformation T0all; // from magnet pipe entrace to origin
70     T0all.translate( Vec3D( 0.0, 0.0, -mag_zero) );
```

```
71
72      Transformation Trot;
73      Trot.rotate_y(defl);
74
75      Transformation Tx = Trot; // applies to imported particle coords TRANSLATION DONT WORK
        FOR SOME REASON, translation applied below
76      Transformation Tv = Trot; // applies to imported particle velocities
77
78
79      /* read geom.dat */
80      std::ifstream is_geom( geom_fn.c_str() );
81      if( !is_geom.good() )//
82          throw( Error( ERROR_LOCATION, (std::string)"couldn\'t open file \'" + geom_fn +
        "\'" ) );
83      Geometry geom( is_geom );
84      is_geom.close();
85      geom.build_surface();
86
87      // init solver
88      EpotBiCGSTABSolver solver( geom, tol ); // solver object (type depends on geometry)
89      solver.set_gnewton( true ); // Enable/disable globally convergent Newton-Raphson (does
        nothing ???) - true is default
90
91      // init efield and space charge
92      EpotField epot( geom );
93      MeshVectorField bfield;
94      MeshScalarField scharge( geom );
95      MeshScalarField scharge_ave( geom ); // for space charge averaging (see radis H-)
96      EpotEfield efield( epot );
97      field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
98                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,  //SYMMETRIC_POTENTIAL
99                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
100     efield.set_extrapolation( efldextrpl );
101
102     // init particle database
103     ParticleDataBase3D pdb( geom );
104     pdb.set_max_steps( 1000 );
105     bool pmirror[6] = { false, false, false, false, false, false };
106     pdb.set_mirror( pmirror );
107     pdb.set_surface_collision( true ); // not in original "Radis", needed or prgram crashes
        when loading a pre-built geom
108
109     // Vlasov iterations
110     for( size_t i = 0; i < iterations; i++ ) { //originally i<15
111          ibsimu.message(1) << "\n --------------------------- " + sfn + " --- Slits ---
        Vlasov loop i = " << i << "\n"; // loop count output
112
113         //solve poisson
114        solver.solve( epot, scharge_ave ); // when SC averaging is removed, put scharge here
115        efield.recalculate();
116         pdb.clear();
117
118         // Input particles
119         ReadAscii din( particles_in_fn, 9 );
120         //cout << "Reading " << din.rows() << " particles\n";
121         for( size_t i = 0; i < din.rows(); i++ ) {
```

```
            double I  = din[0][i];
            double m  = din[1][i];
            double t  = din[2][i];
            double x  = din[3][i];
            double vx = din[4][i];
            double y  = din[5][i];
            double vy = din[6][i];
            double z  = din[7][i];
            double vz = din[8][i];

            Vec3D coord = Tx.transform_vector( Vec3D(x + x_dist - exit_x,
                                                    y,
                                                    z - z_dist - exit_z - mag_zero) );
            Vec3D veloc = Tv.transform_vector( Vec3D(vx, vy, vz) );


            if (filter != 0 && (m/MASS_U < filter-0.1 || m/MASS_U > filter+0.1)){
                continue;
            }
            pdb.add_particle( I, 1.0, m/MASS_U, ParticleP3D(t,coord[0], veloc[0],
                                                            coord[1], veloc[1],
                                                            coord[2] + slit_dist, veloc[2])
    );
        }


        pdb.iterate_trajectories( scharge, efield, bfield ); // error "solid 5 not defined"
    occurs here when pdb.set_surface_collision( false );

        // Space charge averaging (RADIS H-)
       if( i == 0 ) {
           scharge_ave = scharge;
       } else {
            uint32_t nodecount = scharge.nodecount();
            for( uint32_t b = 0; b < nodecount; b++ ) {
                scharge_ave(b) = sc_alpha*scharge(b) + (1.0-sc_alpha)*scharge_ave(b);
            }
       }
    }

    // export
    epot.save( epot_fn );
    pdb.save( pdb_fn );
    scharge_ave.save( scharge_fn );

    // Write output file containing ALL particles 1 mm before slit
    TrajectoryDiagnosticData tdata;
    pdb.trajectories_at_plane(tdata, AXIS_Z, slit_dist-1e-3, {DIAG_CURR, DIAG_MASS, DIAG_T,
                                                              DIAG_X, DIAG_VX, DIAG_Y,
    DIAG_VY, DIAG_Z, DIAG_VZ});
    tdata.export_data(particles_out_ALL_fn);

    // Write output file containing particles that have passed the slit
    ofstream fileOut( particles_out_fn );
    for( size_t k = 0; k < pdb.size(); k++ ) {

```

```
175         Particle3D &pp = pdb.particle( k );
176         // Skip electrons
177         if( pp.m() < 0.5*MASS_U )
178             continue;
179         // Skip ions not at the end
180         if( pp(PARTICLE_Z) < slit_dist + 5e-3 ) // only particles behind slit
181             continue;
182 
183         // Plot particle I, m, coordinates
184         // 3D has 7 coordinates
185         fileOut << setw(12) << pp.IQ() << " ";
186         fileOut << setw(12) << pp.m() << " ";
187         for( size_t j = 0; j < 7; j ++ )
188             fileOut << setw(12) << pp(j) << " ";
189         fileOut << "\n";
190     }
191     fileOut.close();
192 }
193 
194 int main( int argc, char **argv )
195 {
196     try {
197         ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
198        ibsimu.set_thread_count( 4 );
199        simu( argc, argv );
200     } catch( Error e ) {
201        e.print_error_message( ibsimu.message( 0 ) );
202         exit( 1 );
203     }
204 
205     return( 0 );
206 }
```

The analysis utility *analysis.cpp* can be used to generate plots for beam profile, emittance, particle trajectories or electromagnetic field at specified planes. The analysis parameters are set in *master.sh* (line 68-86). The analysis plane is where beam profile and emittance plots are generated. It is perpendicular to *diag_ax_<section>* (lines 78-81) at the coordinate *diag_<section>* in mm. Alternatively, *interact* can be set to 1 (line 69 in *master.sh*) to open an interactive plotter. In this case no plots are exported.

Listing 14: analysis.cpp

```
1  #include <fstream>
2  #include <iomanip>
3  #include <limits>
4  #include <numeric>
5  #include "meshvectorfield.hpp"
6  #include "dxf_solid.hpp"
7  #include "mydxffile.hpp"
8  #include "gtkplotter.hpp"
9  #include "geomplotter.hpp"
10 #include "geometry.hpp"
11 #include "func_solid.hpp"
12 #include "epot_efield.hpp"
```

```
13 #include "error.hpp"
14 #include "ibsimu.hpp"
15 #include "trajectorydiagnostics.hpp"
16 #include "particledatabase.hpp"
17 #include "particlediagplotter.hpp"
18 #include "fielddiagplotter.hpp"
19
20 using namespace std;
21
22 void simu( int argc, char **argv ){
23
24     const int font_size = 15;
25
26     std::string prefix = argv[1]; // filename prefix, e.gh. "ex" for extraction ... ##
27     std::string gfn = argv[2];
28     std::string sfn = argv[3];
29     const bool interactive = atoi(argv[4]); // GTK (1) or geom (0) plotter
30     const char *view = argv[5];
31     const double offs = atof(argv[6]);
32     const bool Jview = atoi(argv[7]); // plot current desity instead of trajectories
33     const bool usrlim = atoi(argv[8]);
34     const double zmin = atof(argv[9]); // axis limts
35     const double zmax = atof(argv[10]); // axis limts
36     const double xymin = atof(argv[11]); // axis limts
37     const double xymax = atof(argv[12]); // axis limts
38     const char *diag_ax = argv[13];
39     const double diag_coord = atof(argv[14]); // distance along beam axis for diagnostics
40     const double r_line = atof(argv[15]); // distance along beam axis for diagnostics
41     const char *efmt = argv[16]; // export fileformat
42     // B params are used directly below
43     //std::string bfn = argv[17]; // bfield data filename (OPTIONAL)
44     //const double bscale = atof(argv[18]); // bfield scaling factor (OPTIONAL)
45     //const double mag_zero = atof(argv[19]); // magnet assembly Z zero (OPTIONAL)
46
47
48     // import/export filenames
49     std::string geom_fn = "dat/" + gfn + "/" + prefix + "_geom_" + gfn + ".dat";
50     std::string epot_fn = "dat/" + gfn + "/" + prefix + "_epot_" + sfn + ".dat";
51     std::string pdb_fn = "dat/" + gfn + "/" + prefix + "_pdb_" + sfn + ".dat";
52     std::string scharge_fn = "dat/" + gfn + "/" + prefix + "_scharge_" + sfn + ".dat";
53
54     /* import geom from build_geom.cpp */
55     std::ifstream is_geom( geom_fn );//
56     if( !is_geom.good() )
57    throw( Error( ERROR_LOCATION, (string)"couldn\'t open file \'" + geom_fn + "\'" ) );
58     Geometry geom( is_geom );
59     is_geom.close();
60     geom.build_surface();
61
62     /* import Epot from simu3.cpp */
63     std::ifstream is_epot( epot_fn );
64     if( !is_epot.good() )
65    throw( Error( ERROR_LOCATION, (string)"couldn\'t open file \'" + epot_fn + "\'" ) );
66     EpotField epot( is_epot, geom );
67     is_epot.close();
68
```

```
69      /* calculate Efield */
70      EpotEfield efield( epot );
71      field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
72                  FIELD_SYMMETRIC_POTENTIAL, FIELD_EXTRAPOLATE,
73                  FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
74      efield.set_extrapolation( efldextrpl );
75
76      /* import particle databse from simu3.cpp */
77      std::ifstream is_pdb( pdb_fn );
78      if( !is_pdb.good() )
79     throw( Error( ERROR_LOCATION, (string)"couldn\'t open file \'" + pdb_fn + "\'" ) );
80      ParticleDataBase3D pdb( is_pdb, geom );
81      is_pdb.close();
82
83      /* import space charge scalar field from simu3.cpp */
84      std::ifstream is_scharge( scharge_fn );
85      if( !is_scharge.good() )
86     throw( Error( ERROR_LOCATION, (string)"couldn\'t open file \'" + scharge_fn + "\'" ) );
87      MeshScalarField scharge( is_scharge );
88      is_scharge.close();
89
90      VectorField *bfield = NULL;
91      if( argc >= 18 ) {
92     bool fout[3] = {true, true, true};
93     MeshVectorField *mesh_bfield = new MeshVectorField( MODE_3D, fout, 1.0e-3,
        atof(argv[18]), argv[17] );
94      mesh_bfield->translate( Vec3D(0.0, 0.0e-3, 1e-3*atof(argv[19]) )); // measured from stl
        file --> yorigin=-395.68
95      field_extrpl_e bfldextrpl[6] = { FIELD_ZERO, FIELD_ZERO,
96                                       FIELD_ZERO, FIELD_ZERO,
97                                       FIELD_ZERO, FIELD_ZERO };
98     mesh_bfield->set_extrapolation( bfldextrpl );
99     bfield = mesh_bfield;
100     }
101
102
103     MeshScalarField tdens( geom );
104     pdb.build_trajectory_density_field( tdens );
105
106     if (interactive){
107         GTKPlotter plotter( &argc, &argv );
108         plotter.set_geometry( &geom );
109         plotter.set_epot( &epot );
110         plotter.set_efield( &efield );
111         if( bfield )
112            plotter.set_bfield( bfield );
113         plotter.set_scharge( &scharge );
114         plotter.set_trajdens( &tdens );
115         plotter.set_particledatabase( &pdb );
116         plotter.new_geometry_plot_window();
117         plotter.run();
118     }else{
119         GeomPlotter geomplotter( geom );
120         geomplotter.set_epot( &epot );
121         geomplotter.set_efield( &efield );
122         if( bfield )
```

```
123             geomplotter.set_bfield( bfield );
124         geomplotter.set_scharge( &scharge );
125         geomplotter.set_trajdens( &tdens );
126         geomplotter.set_particledatabase( &pdb );
127         geomplotter.set_qm_discretation(true); // plot different masses in different colors
128         geomplotter.set_eqlines_manual( {2, .3, -1, -5, -20} );
129 
130         // view
131         if ( ! strcmp(view, "zx") ){
132             geomplotter.set_view_si(VIEW_ZX, offs); // second arg can be usedas offset
    along third axis
133         } else if ( ! strcmp(view, "zy")  ){
134             geomplotter.set_view_si(VIEW_ZY, offs );
135         } else if ( ! strcmp(view, "xy") ) {
136             geomplotter.set_view_si(VIEW_XY, offs );
137         }
138 
139         // current density plt
140         if (Jview){
141             geomplotter.set_particle_div(0); // no particles
142             geomplotter.set_fieldgraph_plot(FIELD_SCHARGE); // formerly FIELD_TRAJDENS, but
    has bad interpolation within magnet
143             FieldGraph *fgraph = geomplotter.fieldgraph();
144             fgraph->set_zscale(ZSCALE_RELLOG); // LOG can also be used, but z_min and z_max
    have to be set manually (see FieldGraph)
145         }
146 
147         // axis lim
148         if (usrlim){ // if not, auto-set (full range)
149             geomplotter.set_ranges(zmin, xymin, zmax, xymax); // xmin, ymin, xmax, ymax
150         }
151 
152         // ######## EMITTANCE & PROFILE PLOTS ##############
153         // emittance x
154         ParticleDiagPlotter pdiagplotterx( geom, pdb, AXIS_Z, diag_coord,
    PARTICLE_DIAG_PLOT_HISTO2D, DIAG_X, DIAG_XP);
155         if ( ! strcmp(diag_ax, "x") ){
156             pdiagplotterx.set_view(AXIS_X, diag_coord);
157             pdiagplotterx.set_plot(PARTICLE_DIAG_PLOT_HISTO2D, DIAG_Z, DIAG_ZP); // plot z
    emittance if diag is in x dir
158         } else if ( ! strcmp(diag_ax, "y") ){
159             pdiagplotterx.set_view(AXIS_Y, diag_coord); // makes no sense in y
160         }
161         pdiagplotterx.calculate_emittance();
162         pdiagplotterx.set_emittance_ellipse ( true );
163         // emittance y
164         ParticleDiagPlotter pdiagplottery( geom, pdb, AXIS_Z, diag_coord,
    PARTICLE_DIAG_PLOT_HISTO2D, DIAG_Y, DIAG_YP);
165         if ( ! strcmp(diag_ax, "x") ){
166             pdiagplottery.set_view(AXIS_X, diag_coord);
167         } else if ( ! strcmp(diag_ax, "y") ){
168             pdiagplottery.set_view(AXIS_Y, diag_coord);
169         }
170         pdiagplottery.calculate_emittance();
171         pdiagplottery.set_emittance_ellipse ( true );
172 
```

```
173         // profile
174         ParticleDiagPlotter pdiagplotterp( geom, pdb, AXIS_Z, diag_coord,
    PARTICLE_DIAG_PLOT_HISTO2D, DIAG_X, DIAG_Y);
175         std::vector<Particle3D> tdata; // find out maximum x or y coordingate in pdb to set
    x,y plot range to the same value; avoid distorted display (more below)
176         for( size_t k = 0; k < pdb.size(); k++ ) {
177             tdata.push_back( pdb.particle( k ) );
178         }
179         std::vector<double> xvals;
180         std::vector<double> yvals;
181         if ( ! strcmp(diag_ax, "x") ){
182             pdiagplotterp.set_view(AXIS_X, diag_coord);
183             pdiagplotterp.set_plot(PARTICLE_DIAG_PLOT_HISTO2D, DIAG_Z, DIAG_Y);
184             pdb.trajectories_at_plane( tdata, AXIS_X, diag_coord );
185             for(unsigned int i=0; i < tdata.size(); i++){
186                 Particle3D &pp = tdata[i];
187                 double xi = pp(PARTICLE_Z);
188                 double yi = pp(PARTICLE_Y);
189                 xvals.push_back(xi);
190                 yvals.push_back(yi);
191             }
192         } else if ( ! strcmp(diag_ax, "y") ){
193             pdiagplotterp.set_view(AXIS_Y, diag_coord);
194             pdiagplotterp.set_plot(PARTICLE_DIAG_PLOT_HISTO2D, DIAG_X, DIAG_Z);
195             pdb.trajectories_at_plane( tdata, AXIS_Y, diag_coord );
196             for(unsigned int i=0; i < tdata.size(); i++){
197                 Particle3D &pp = tdata[i];
198                 double xi = pp(PARTICLE_X);
199                 double yi = pp(PARTICLE_Z);
200                 xvals.push_back(xi);
201                 yvals.push_back(yi);
202             }
203         } else if ( ! strcmp(diag_ax, "z") ){
204             pdb.trajectories_at_plane( tdata, AXIS_Z, diag_coord );
205             for(unsigned int i=0; i < tdata.size(); i++){
206                 Particle3D &pp = tdata[i];
207                 double xi = pp(PARTICLE_X);
208                 double yi = pp(PARTICLE_Y);
209                 xvals.push_back(xi);
210                 yvals.push_back(yi);
211             }
212         }
213         double avgx = accumulate( xvals.begin(), xvals.end(), 0.0)/xvals.size(); // x avg
214         double avgy = 0.0; // y is mirrored--> force to 0 //accumulate( yvals.begin(),
    yvals.end(), 0.0)/yvals.size(); // y avg
215         double maxx = 0;
216         double maxy = 0;
217         for(unsigned int i=0; i < xvals.size(); i++){
218             double xia = abs( xvals[i] );
219             double yia = abs( yvals[i] );
220             if ( xia > maxx )
221                 maxx = xia;
222             if ( yia > maxy )
223                 maxy = yia;
224         }
225         double diffx = maxx - avgx;
```

```
226         double diffy = maxy - avgy;
227         double diff; // one diff fo quadratic plot
228         if (diffx > diffy) {
229             diff = diffx;
230         } else {
231             diff = diffy;
232         }
233         diff *= 1.1; // add 10% to plot range
234         pdiagplotterp.set_ranges(avgx-diff, avgy-diff, avgx+diff, avgy+diff);
235 
236         // ##### EXPORT ############
237         int png_res = 3000; // glaube das wird automatisch freigestellt
238         pdiagplotterx.set_font_size(font_size);
239         pdiagplottery.set_font_size(font_size);
240         pdiagplotterp.set_font_size(font_size);
241         if ( ! strcmp(efmt, "png") ){
242             geomplotter.set_size(png_res,png_res);
243             geomplotter.set_font_size(30);
244             geomplotter.plot_png( "fig/" + gfn + "/" + prefix + "_traj_" + sfn + ".png" );
245             pdiagplotterx.plot_png( "fig/" + gfn + "/" + prefix + "_emitx_" + sfn + ".png"
    );
246             pdiagplottery.plot_png( "fig/" + gfn + "/" + prefix + "_emity_" + sfn + ".png"
    );
247             pdiagplotterp.plot_png( "fig/" + gfn + "/" + prefix + "_profile_" + sfn +
    ".png" );
248         } else if ( ! strcmp(efmt, "pdf") ) {
249             geomplotter.plot_pdf( "fig/" + gfn + "/" + prefix + "_traj_" + sfn + ".pdf" );
250             pdiagplotterx.plot_pdf( "fig/" + gfn + "/" + prefix + "_emitx_" + sfn + ".pdf"
    );
251             pdiagplottery.plot_pdf( "fig/" + gfn + "/" + prefix + "_emity_" + sfn + ".pdf"
    );
252             pdiagplotterp.plot_pdf( "fig/" + gfn + "/" + prefix + "_profile_" + sfn +
    ".pdf" );
253         } else if ( ! strcmp(efmt, "svg") ) {
254             geomplotter.plot_svg( "fig/" + gfn + "/" + prefix + "_traj_" + sfn + ".svg" );
255             pdiagplotterx.plot_svg( "fig/" + gfn + "/" + prefix + "_emitx_" + sfn + ".svg"
    );
256             pdiagplottery.plot_svg( "fig/" + gfn + "/" + prefix + "_emity_" + sfn + ".svg"
    );
257             pdiagplotterp.plot_svg( "fig/" + gfn + "/" + prefix + "_profile_" + sfn +
    ".svg" );
258         }
259 
260 
261         // space charge and potential in line cross section; extraction only
262         const char *pref_char = argv[1]; // sama as 'prefix', but as char array
263         if (!strcmp(pref_char, "ex")){
264 
265             double z_section = -20.0e-3; // z coord of cross section
266             int steps = 100;
267             double xrange = r_line + 1.0e-3;
268             Vec3D diag_start( -xrange, 0.0, z_section ); // (x,y,z)
269             Vec3D diag_stop( +xrange, 0.0, z_section ); // (x,y,z)
270             field_diag_type_e diag_y[2] = {FIELD_EPOT, FIELD_SCHARGE};
271             field_loc_type_e diag_x_axis[2] = {FIELDD_LOC_X, FIELDD_LOC_X};
272 
```

```
273             FieldDiagPlotter fdplotter( geom );
274             fdplotter.set_epot( &epot );
275             fdplotter.set_efield( &efield );
276             if( bfield )
277                fdplotter.set_bfield( bfield );
278             fdplotter.set_scharge( &scharge );
279             fdplotter.set_trajdens( &tdens );
280             fdplotter.set_coordinates(steps, diag_start, diag_stop);
281             fdplotter.set_diagnostic(diag_y, diag_x_axis); // plot x on x-axis
282 
283             // output
284             png_res = 1000;
285             fdplotter.set_size(png_res,png_res);
286             fdplotter.set_font_size(20);
287             fdplotter.plot_png( "fig/" + gfn + "/" + prefix + "_line_sect_" + sfn + ".png"
    );
288         }
289     }
290 
291 }
292 
293 
294 int main( int argc, char **argv )
295 {
296 
297     try {
298        ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
299        ibsimu.set_thread_count( 4 );
300        simu( argc, argv );
301     } catch( Error e ) {
302        e.print_error_message( ibsimu.message( 0 ) );
303         exit( 1 );
304     }
305 
306     return( 0 );
307 }
```

The utility *analysis.py* is only applied to particles of the slit section simulation (line 134 in *master.sh*). It generates plots similar to Fig. 15. Note that this requires a Python 3 environment.

Listing 15: analysis.py

```
1  from matplotlib import pyplot as plt
2  from matplotlib.ticker import EngFormatter
3  from source import tools as t
4  import numpy as np
5  import sys
6  import csv
7  from scipy import constants as const
8 
9  fn = sys.argv[1] # filename prefix
10 gfn = sys.argv[2] # geometry filename
11 I0 = sys.argv[3] # initial current
12 slit_dist = 1e-3*float(sys.argv[4])
```

```
slit_sep = 1e-3*float(sys.argv[5])
bscale = sys.argv[6] # scaling factor for B field
bmax = 13.33332112108444 # B at approx 0,0,0 in model (not mapping!) data (dpace 1.24749)
b = bmax*float(bscale) # B in T
io_particles_fn = 'dat/' + gfn + '/io_particles_' + fn + '_I' + I0 + '.txt' # particle data in front of magnet
sl_particles_all_fn = 'dat/' + gfn + '/sl_particles_ALL_' + fn + '_I' + I0 + '.txt' # particle data in front of slit
sl_particles_fn = 'dat/' + gfn + '/sl_particles_' + fn + '_I' + I0 + '.txt' # particle data after slit
mass_spec_fn = 'dat/' + gfn + '/mass_spectrum_' + fn + '_I' + I0 + '.txt'
mass_spec_fig_fn = 'fig/' + gfn + '/mass_spectrum_' + fn + '_I' + I0 + '.png'
xsect_fig_fn = 'fig/' + gfn + '/xsect_' + fn + '_I' + I0 + '.png'
mcomp_fig_fn = 'fig/' + gfn + '/mcomp_' + fn + '_I' + I0 + '.png'

################### READ PARTICLE DATA ##########################
# import particle data in front of magnet
try:
    I_all, m_all, t_all, x_all, vx_all, y_all, vy_all, z_all, vz_all = np.loadtxt(sl_particles_all_fn, skiprows = 0, unpack = True) # import particles
except ValueError:
    I_all, m_all, t_all, x_all, vx_all, y_all, vy_all, z_all, vz_all = [np.zeros(1)] * 9
    print("WARNING: empty particle data input (no particles in front of slit)")

# calculate total current I_io_tot and current for chosen masses in I_io_mass
I_all_tot = np.sum(I_all) # total current in front of slit
masses = [149, 150, 152]
I_all_mass = [0] * len(masses) # current per mass
for i in range(len(I_all)): # iterate through data rows
    for j, m in enumerate(masses): # iterate through chosen masses
        if abs(m_all[i] - m) < 0.1: # 0.1 u tolerance
            I_all_mass[j] = I_all_mass[j] + I_all[i] # add to mass specific total

# import particle data after slit
try:
    I_sl, m_sl, t_sl, x_sl, vx_sl, y_sl, vy_sl, z_sl, vz_sl = np.loadtxt(sl_particles_fn, skiprows = 0, unpack = True) # import particles
except ValueError:
    I_sl, m_sl, t_sl, x_sl, vx_sl, y_sl, vy_sl, z_sl, vz_sl = [np.zeros(1)] * 9
    print("WARNING: empty particle data input (no particles after slit)")
I_sl_tot = np.sum(I_sl) # transmitted current
m_sl = [mi / const.u for mi in m_sl]

pcharge = I_all[0] #charge of one macroparticle

################### GLOBAL PLOT SETTINGS #############################
title = "Slit dist. = " + "{:.3f}".format(slit_dist) + " m\n" + r"$B = $" + "{:.4f}".format(b) + " T\n"
for j, m in enumerate(masses):
    title = title + "\n" + r'$I_{tot}($' + str(m) + r' u$)$ = ' + t.eng_string(float(I_all_mass[j]), format="%.2f", si=True) + 'A'
title = title + "\n" + r"$I_\mathrm{T} = $"  + t.eng_string(I_sl_tot, format="%.2f", si=True) + 'A'

fs = 12
axislabel_fontsize = fs
```

```
61
62  formatter1 = EngFormatter(places=0, sep="\N{THIN SPACE}")  # U+2009
63
64  ################## X SECTION SPECTRUM PLOT ########################
65  fig0, ax0 = plt.subplots()
66  n, bins, patches = ax0.hist(x_all, bins = 100, weights = I_all, align = "mid", label =
        "Beam at slit")
67  ax0.hist(x_sl, bins = bins, align = "mid", weights = I_sl, alpha=1, label = "Beam 10 mm
        behind slit")
68
69  ax0.axvline(slit_sep/2, color = 'k', label = "Slit limits (" + str(slit_sep*1e3) +" mm)")
70  ax0.axvline(-slit_sep/2, color= 'k')
71
72  ax0.set_yscale('log')
73
74  ax0.grid(visible=True, which='major', color='grey', linestyle='-', lw=0.5)
75  ax0.grid(visible=True, which='minor', color='grey', axis='y', linestyle='-', lw=0.25)
76
77  t.scale_axis(1e-3, "x", ax0)
78  ax0.yaxis.set_major_formatter(formatter1)
79
80  ax0.set_xlabel(r'Horizontal offset (mm)', fontsize = axislabel_fontsize) # x label
81  ax0.set_ylabel(r'Ion current / bin (A)', fontsize = axislabel_fontsize) # y label
82
83  ax0.set_title(title, loc = 'left', fontsize = fs)
84  ax0.legend(loc = 'upper right', bbox_to_anchor=(1.0, 1.26), fontsize = fs)
85
86  fig0.savefig(xsect_fig_fn, bbox_inches = 'tight', format = 'png', transparent=False)
87
88  ################## MASS COMPOSITION PLOT ########################
89  fig1, ax1 = plt.subplots()
90  mbins = np.arange(np.min(masses)-0.5, np.max(masses)+0.51, 1)
91
92  ax1.hist((m_all,m_sl), bins = mbins, weights = (I_all,I_sl), align = "mid", label = ("Beam
        at slit","Beam 10 mm behind slit"), ec='black')
93  #ax1.hist(m_sl, bins = mbins, align = "mid", weights = I_sl, alpha=.7, label = "Beam 10 mm
        behind slit")
94
95  ax1.set_ylim(bottom=pcharge/2)
96  ax1.set_yscale('log')
97
98  ax1.grid(visible=True, which='major', color='grey', linestyle='-', lw=0.5)
99  ax1.grid(visible=True, which='minor', color='grey', axis='y', linestyle='-', lw=0.25)
100
101 ax1.yaxis.set_major_formatter(formatter1)
102 ax1.set_xticks(np.arange(np.min(masses), np.max(masses)+0.1, 1))
103 ax1.set_xlabel(r'Mass (u)', fontsize = axislabel_fontsize) # x label
104 ax1.set_ylabel(r'Ion current / mass (A)', fontsize = axislabel_fontsize) # y label
105
106 ax1.set_title(title, loc = 'left', fontsize = fs)
107 ax1.legend(loc = 'upper right', bbox_to_anchor=(1.0, 1.22), fontsize = fs)
108
109 fig1.savefig(mcomp_fig_fn, bbox_inches = 'tight', format = 'png', transparent=False)
110
111 ################## MASS SPECTRUM PLOT ########################
112 # export mass scan data (appends to list; clear or rename list if filename already exists)
```

```
if False:
    with open(mass_spec_fn, "a", newline='') as file:
        writer = csv.writer(file, delimiter = '\t')
        writer.writerow([b, I_sl_tot])
    file.close()
'''
try:
    b_in, I_in = np.loadtxt(mass_spec_fn, unpack=True)
except ValueError:
    b_in, I_in = [np.zeros(1)] * 2
    print("WARNING: empty mass spectrum data. Plotting zeros.")


fig2, ax2 = plt.subplots()
ax2.step(b_in, I_in)

ax2.set_yscale('log')
ax2.yaxis.set_major_formatter(formatter1)

ax2.grid(visible=True, which='major', color='grey', linestyle='-', lw=0.5)
ax2.grid(visible=True, which='minor', color='grey', axis='y', linestyle='-', lw=0.25)

ax2.set_xlabel(r'Magnetic Field /T', fontsize = axislabel_fontsize) # x label
ax2.set_ylabel(r'Ion current /A', fontsize = axislabel_fontsize) # y label

ax2.set_title(title, loc = 'left', fontsize = fs)

fig2.savefig(mass_spec_fig_fn, bbox_inches = 'tight', format = 'pdf', transparent=False)
'''
```

The library *plasma.h* implements some functions to calculate the electric potential of a thermal plasma according to [45]. It can be used by setting *use_analytic_plasma* to 1 (Line 52 in *master.sh*). However, the simulation seems to encounter convergence problems in this case and it is therefore recommended to use the IBSimu builtin plasma model, although this has its own caveats (see Sec. 3.3.1).

Listing 16: plasma.h

```
double c_kb = 1.380649e-23; // Boltzmann in J/K
double c_e = 1.602176634e-19; // elementary charge in C
double c_h = 6.62607015e-34; // plancks const in Js
double c_e0 = 8.8541878128e-12; // electric field constant in As/Vm
double c_me = 9.1093837015e-31; // e- mass in kg
double c_pi = 3.141592653589793; // pi

double where(bool cond, double def, double x){
    // similar to numpy.where; returns default where cond is true, else x
    if (cond){
        return def;
    }else{
        return x;
    }
}

```

```
double richardson(double W, double T) {
    // electron density (in 1/m^3) on surface according to Richardson law
    return 2*pow( ( 2*c_pi*c_me*c_kb*T/(c_h*c_h) ),(1.5) ) * exp( -c_e*W/(c_kb*T) );
}

double deybe(double n_p, double T){
    // deybe shielding distance in m
    return sqrt( c_e0*c_kb*T/(n_p*c_e*c_e) );
}

double phi(double x, double W, double T, double n_p){
    // from Lawson, CERN report 76-09, plasma in a box
    x = where(x<0, 0, x); // fix x<0 to x=0 to avoid invalid input
    double n_e0 = richardson(W, T);
    double l_D = deybe(n_p, T);
    double A = 4*c_kb*T/c_e;
    double B = sqrt(2)/l_D;
    double C = log(n_e0/n_p);
    double x0 = - log( tanh( C/4 ) )/B; // solved potential eq. with 4*phi(0)=ln(n_e0/n_p)
    for integrtion contant x0 (wolfram alpha)
    double phi = A * atanh( exp( -(x+x0)*B ) );
    double phi0 = A * atanh( exp( -(0.0+x0)*B ) );
    return phi - phi0;
}

double phi_cav(double x, double y, double W, double T, double n_p, double d){
    double r = sqrt(x*x + y*y);
    r += d/2; // move 0 to mid of line instead of wall
    r = where(r>d, d, r); // fix r>d to r=d to avoid invalid input
    double n_e0 = richardson(W, T);
    double l_D = deybe(n_p, T);
    double A = 4*c_kb*T/c_e;
    double B = sqrt(2)/l_D;
    double C = log(n_e0/n_p);
    // from wolfram alpha: solve arctanh(exp(-x0*B)) + arctanh(exp(-(d+x0)*B)) = C/4 for x0
    double a1 = 1/( 2*(exp(C)+1) );
    double b1 = pow( (-2*exp(C/2 - B*d) - exp(C - B*d) - exp(-B*d) - 2*exp(C/2) - exp(C) -
    1), 2.0);
    double c1 = 4*(1 - exp(C)) * (exp(C) - 1) * exp(-B*d);
    double d1 = 2*exp(C/2 - B*d) + exp(C - B*d) + exp(-B*d) + 2*exp(C/2) + exp(C) + 1;
    double x0 = (1/B) * log( a1 * ( sqrt(b1 + c1) + d1 ));
    double phi = A*(atanh( exp( -(r+x0)*B ) ) +
                    atanh( exp( -(d-r+x0)*B ) ) );
    double phi0 = A*(atanh( exp( -(0.0+x0)*B ) ) +
                    atanh( exp( -(d-0.0+x0)*B ) ) );
    return phi - phi0;
}
```

6.0.1 IBSimu installation guide

IBSimu Installation

Ion Beam Simulator (IBSimu) is a C++ library, which means that it is not a „ready-to-use“ software, but is distributed as source code and can be compiled and used with code written by the user. This allows great flexibility, for example the code can be modified to the users needs or the user programs can be compiled and run on MOGON. However, for windows users the installation of the library is everything else than straight forward, so this document serves as addition to the installation instructions of the IBSimu homepage. Keep in mind that this guide is written by a non-linux-user.

Prerequisites

The tools needed for IBSimu (C++ compiler, linker, make, pkg-config) are already installed on most linux systems, otherwise refer to the respective package manager manuals on how to install packages and libraries on your distribution. IBSimu has been successfully tested on MOGON, so all software needed is available on MOGON as well.

To run IBSimu on Windows, the installation of Linux (for example Xubuntu) in a VirtualBox is recommeded. It seems that, altough mentioned in the IBSimu installation instructions, the installation of IBSimu with MSYS2 directly in windows is not working, because IBSimu or one of its dependencies requires linux-only header files. However, the installation of the VirtualBox is simple and works well. Make sure that you install a 64-bit system; if VirtualBox does not offer the possibility to install a 64-bit system, then *64-bit virtualization* has to be enabled in your computers BIOS. **Note that Windows 10 features a linux subsystem (WSL)**, which can be installed from the Microsoft store and is perfectly capable of runnig IBsimu. This might be the most straightforward option for Windows 10 users.

By now the Windows WSL verison 2 is released. It seems the interactive plotter is not working there because XServer forwarding is not straightforward. Reverting to WSL Version 1 fixes the problem. Related: https://askubuntu.com/questions/1299323/how-to-set-up-display-variable-for-wsl2-of-ubuntu-20

Installation

Package download and installation

This installation guide refers to IBSimu version 1.0.6 (May 2017).

Update package manager

```
sudo apt update
```

Install gcc, g++, make and pkg-config:

```
sudo apt install build-essential
sudo apt-get install pkg-config
```

Install required libraries:

```
sudo apt install -y zlib1g-dev \
libpng-dev \
libfreetype6-dev \
libfontconfig1-dev \
libcairo2-dev \
libclthreads-dev \
libgsl-dev \
libgtk-3-dev \
libgtkglextmm-x11-1.2-dev \
libumfpack5 \
libsuitesparse-dev \
libopencsg-dev
```

Download the most recent version of IBSimu in your `src` (in home directory).

```
mkdir src
cd src
wget -O libibsimu-1.0.6.tar.gz
https://sourceforge.net/projects/ibsimu/files/latest/download
```

Then unpack the downloaded archive:

```
tar -zxvf libibsimu-1.0.6.tar.gz
```

and change into the IBSimu folder:

```
cd libsimu-1.0.6
```

Perform configuration and compilation:

```
./configure --prefix=/home/<username>
make
make check
make install
```

Here <username> is your username. Refer to the troubleshoot below when problems occured during one of these steps. On successful installation you should have two new entries in your home directory:

```
/include
+  /libibsimu-1.0.6dev
   +  config.h
   +  *.hpp
```

```
/lib
+  libibsimu-1.0.6.a
+  libibsimu-1.0.6.1a
+  libibsimu-1.0.6.so
+  libibsimu-1.0.6.so.0
+  libibsimu-1.0.6.so.0.0.1
+  /pkgconfig
   +  ibsimu-1.0.6.pc
```

Path variables

To successfully compile your own code with IBSimu, the compiler has to be informed about the locations of the IBSimu files, for example if you want to `#include „an_ibsimu_file.hpp“`, it has to search the location where it is stored to find it. Furthermore the compilation has to be run with specific compiler options, which is managed by the helper tool `pkg-config`. This tool has to know the location of the config file `ibsimu-1.0.6.pc`. To specify the search paths for libraries and includes, environment variables are used. If you call the compiler, these variables can be used as flags, for example with `Make` or `CMake`. To set the variables, you can use

```
export PATH="$PATH:/home/<username>/lib"
```

but the variable will be reset for every new session. To have the variables set every time you log in, simply add these lines to the `.profile` file in your home directory:

```
export PATH="${HOME}/bin:${HOME}/lib:${PATH}"
export LDFLAGS="-L${HOME}/lib"
export PKG_CONFIG_PATH="${HOME}/lib/pkgconfig"
export LD_LIBRARY_PATH="${HOME}/lib"
```

`.profile` will be run automatically every time you log in. You have to log out and in again to use the new path variables.

If you are working on Ubuntu within Windows 10, you have to redirect the display to Windows using

```
export DISPLAY=:0
```

You can set this permanently by adding this line to .profile. An **X11 server in Windows is required** (XMing or similar)

Installation trouble shoot

./configure halts with an error, because a package or a library is not installed

Try to access the program via command line or identify the path to the library via `whereis <libraryname>`. On your **own machine**, simply install a missing package. On Ubuntu this is done with

```
sudo apt-get install <that-missing-package-./configure-complains-about>
```

If you cannot access a package on **MOGON**, try to load the module via `module load <modulename>`. Check the environment module documentation for more information.

./configure halts with an error, because `zlib` is not installed, though I installed it moments before

You always need the development package (`<package-name>-devel`) installed on your system. Sometimes, the development package has a different name, for example `zlib1g-devel`. On Ubuntu you can have an overview over all installable packages with

```
apt list <modulename>
```

Usage

Check installation

To check installation, simply download a tutorial file from the homepage. Do not forget its corresponding `Makefile`. Having both files in the same current folder, simply execute

```
make
./<tutorialname>
```

to compile and run the source code. In case one of the steps does not work refer to the troubleshoot section below.

Make

Makefiles will help you to manage all the compiler flags and communicate the library paths with the compiler. For the tutorials example `Makefiles` are available. The content might look like this:

```
 CC = g++
 LDFLAGS = `pkg-config --libs ibsimu-1.0.6dev`
 CXXFLAGS = -Wall -g `pkg-config --cflags ibsimu-1.0.6dev`
 vlasov2d: vlasov2d.o
$(CC) -o vlasov2d vlasov2d.o $(LDFLAGS)
 vlasov2d.o: vlasov2d.cpp
$(CC) -c -o vlasov2d.o vlasov2d.cpp $(CXXFLAGS)
 clean:
$(RM) *~ *.o vlasov2d
```

For the LDFLAGS the helper tool `pkg-config` will be called with arguments `–libs ibsimu-1.0.6dev`, which will return all package dependencies for `ibsimu-1.0.6dev`. For CXXFLAGS the tool will be called for `–cflags ibsimu-1.0.6dev`, which returns all the lookup paths for the packages required.

CMake

When working with CLion as IDE (available for free for students) CMake will be used instead of make. CMake is available on MOGON as well, so it might be used without loss of functionality. CMake seems to be a lot more powerful, but is also more difficult to use. The following lines will call pkg-config with arguments for library dependencies and lookup paths in a similar way as above for the use of make:

```
 exec_program(pkg-config ARGS --libs ibsimu-1.0.6dev OUTPUT_VARIABLE
IBSIMU_LIBS)
 exec_program(pkg-config ARGS --cflags ibsimu-1.0.6dev OUTPUT_VARIABLE
IBSIMU_FLAGS)
 target_link_libraries(<project_name> ${IBSIMU_LIBS})
 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${IBSIMU_FLAGS}")
```

Usage troubleshooting

The code does not compile, because libraries are not found

Check, if your path variables are set correctly:

```
 echo $PATH
 echo $PKG_LIBRARY_CONFIG
 echo $LD_LIBRARY_PATH
```

If these are not set correctly, modify your .profile file in your home directory as explained in the installation part of this document. Furthermore check the pkg-config calls of your make or CMake file respectively:

```
 pkg-config --libs ibsimu-1.0.6dev
 pkg-config --cflags ibsimu-1.0.6dev
```

Links and further information

- http://ibsimu.sourceforge.net/index.html Official homepage
- http://ibsimu.sourceforge.net/download.html Download of IBSimu
- http://ibsimu.sourceforge.net/installation.html Installation instructions, together with library dependencies

References

[1] R. dos Santos Augusto, L. Buehler, Z. Lawson, S. Marzari, M. Stachura, T. Stora, C.-M. collaboration, *CERN-MEDICIS (Medical Isotopes Collected from ISOLDE): A New Facility*, Applied Sciences **4**(2), 265–281 (2014). doi:10.3390/app4020265, URL http://www.mdpi.com/2076-3417/4/2/265

[2] M. Ayranov, D. Schumann, *Preparation of 26Al, 59Ni, 44Ti, 53Mn and 60Fe from a proton irradiated copper beam dump*, Journal of Radioanalytical and Nuclear Chemistry **286**(3), 649–654 (2010). doi:10.1007/s10967-010-0732-0, URL http://link.springer.com/10.1007/s10967-010-0732-0

[3] F. Mertes, N. Kneip, R. Heinke, T. Kieck, D. Studer, F. Weber, S. Röttger, A. Röttger, K. Wendt, C. Walther, *Ion implantation of 226Ra for a primary 222Rn emanation standard*, Applied Radiation and Isotopes **181**, 110093 (2022). doi:10.1016/j.apradiso.2021.110093, URL https://linkinghub.elsevier.com/retrieve/pii/S0969804321004838

[4] N. Kneip, D. Studer, T. Kieck, J. Ulrich, R. Dressler, D. Schumann, K. Wendt, *Separation of Manganese Isotopes by Resonance Ionization Mass Spectrometry for ^{53}Mn Half-Life Determination*, The European Physical Journal Applied Physics (2022). doi:10.1051/epjap/2022210270, URL https://www.epjap.org/10.1051/epjap/2022210270

[5] T. Kieck, H. Dorrer, C. E. Düllmann, V. Gadelshin, F. Schneider, K. Wendt, *Highly efficient isotope separation and ion implantation of 163Ho for the ECHo project*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **945**, 162602 (2019). doi:10.1016/j.nima.2019.162602, URL https://linkinghub.elsevier.com/retrieve/pii/S016890021931109X

[6] Y. Martinez Palenzuela, *Characterization and optimization of a versatile laser and electron-impact ion source for radioactive ion beam production at ISOLDE and MEDICIS*, Ph.D. thesis, KU Leuven (2019). URL http://cds.cern.ch/record/2672954/files/CERN-THESIS-2019-032.pdf?version=1

[7] M. J. Dresser, *The Saha-Langmuir Equation and its Application*, Journal of Applied Physics **39**(1), 338–339 (1968). doi:10.1063/1.1655755, URL http://aip.scitation.org/doi/10.1063/1.1655755

[8] R. Kirchner, *Progress in ion source development for on-line separators*, Nuclear Instruments and Methods in Physics Research **186**(1-2), 275–293 (1981). doi:10.1016/0029-554X(81)90916-2, URL https://linkinghub.elsevier.com/retrieve/pii/0029554X81909162

[9] R. Kirchner, *On the thermoionization in hot cavities*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated

Equipment **292**(2), 203–208 (1990). doi:10.1016/0168-9002(90)90377-I, URL https://linkinghub.elsevier.com/retrieve/pii/016890029090377I

[10] M. Manzolaro, F. D'Agostini, A. Monetti, A. Andrighetto, *The SPES surface ionization source*, Review of Scientific Instruments **88**(9), 093302 (2017). doi:10.1063/1.4998246, URL http://aip.scitation.org/doi/10.1063/1.4998246

[11] CERN-ISOLDE RILIS team, *RILIS Elements* (2013). URL https://riliselements.web.cern.ch/

[12] U. Köster, *Resonance ionization laser ion sources*, Nuclear Physics A **701**(1-4), 441–451 (2002). doi:10.1016/S0375-9474(01)01625-6, URL https://linkinghub.elsevier.com/retrieve/pii/S0375947401016256

[13] B. A. Marsh, *Resonance ionization laser ion sources for on-line isotope separators*, Review of Scientific Instruments **85**(2), 02B923 (2014). doi:10.1063/1.4858015, URL http://aip.scitation.org/doi/10.1063/1.4858015

[14] V. Fedosseev, K. Chrysalidis, T. D. Goodacre, B. Marsh, S. Rothe, C. Seiffert, K. Wendt, *Ion beam production and study of radioactive isotopes with the laser ion source at ISOLDE*, Journal of Physics G: Nuclear and Particle Physics **44**(8), 084006 (2017). doi:10.1088/1361-6471/aa78e0, URL https://iopscience.iop.org/article/10.1088/1361-6471/aa78e0

[15] L. Penescu, R. Catherall, J. Lettry, T. Stora, *Development of high efficiency Versatile Arc Discharge Ion Source at CERN ISOLDE*, Review of Scientific Instruments **81**(2), 02A906 (2010). doi:10.1063/1.3271245, URL http://aip.scitation.org/doi/10.1063/1.3271245

[16] T. Day Goodacre, J. Billowes, R. Catherall, T. Cocolios, B. Crepieux, D. Fedorov, V. Fedosseev, L. Gaffney, T. Giles, A. Gottberg, K. Lynch, B. Marsh, T. Mendonça, J. Ramos, R. Rossel, S. Rothe, S. Sels, C. Sotty, T. Stora, C. Van Beveren, M. Veinhard, *Blurring the boundaries between ion sources: The application of the RILIS inside a FEBIAD type ion source at ISOLDE*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **376**, 39–45 (2016). doi:10.1016/j.nimb.2016.03.005, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X16002111

[17] Y. Martinez Palenzuela, B. Marsh, J. Ballof, R. Catherall, K. Chrysalidis, T. Cocolios, B. Crepieux, T. Day Goodacre, V. Fedosseev, M. Huyse, P. Larmonier, J. Ramos, S. Rothe, J. Smith, T. Stora, P. Van Duppen, S. Wilkins, *Enhancing the extraction of laser-ionized beams from an arc discharge ion source volume*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **431**, 59–66 (2018). doi:10.1016/j.nimb.2018.06.006, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X18303823

[18] K. Zimmer, *Konzeption, Aufbau und Test der Ionenoptik des RISIKO Massenseparators*, Diploma thesis, Mainz (1990).

[19] K. Zhang, D. Studer, F. Weber, V. M. Gadelshin, N. Kneip, S. Raeder, D. Budker, K. Wendt, T. Kieck, S. G. Porsev, C. Cheung, M. S. Safronova, M. G. Kozlov, *Detection of the Lowest-Lying Odd-Parity Atomic Levels in Actinium*, Physical Review Letters **125**(7), 073001 (2020). doi:10.1103/PhysRevLett.125.073001, URL https://link.aps.org/doi/10.1103/PhysRevLett.125.073001

[20] D. Studer, J. Ulrich, S. Braccini, T. S. Carzaniga, R. Dressler, K. Eberhardt, R. Heinke, U. Köster, S. Raeder, K. Wendt, *High-resolution laser resonance ionization spectroscopy of* $^{143-147}Pm$, The European Physical Journal A **56**(2), 69 (2020). doi:10.1140/epja/s10050-020-00061-8, URL http://link.springer.com/10.1140/epja/s10050-020-00061-8

[21] F. Schneider, K. Chrysalidis, H. Dorrer, C. Düllmann, K. Eberhardt, R. Haas, T. Kieck, C. Mokry, P. Naubereit, S. Schmidt, K. Wendt, *Resonance ionization of holmium for ion implantation in microcalorimeters*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **376**, 388–392 (2016). doi:10.1016/j.nimb.2015.12.012, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X15012562

[22] T. Kieck, S. Biebricher, C. E. Düllmann, K. Wendt, *Optimization of a laser ion source for 163 Ho isotope separation*, Review of Scientific Instruments **90**(5), 053304 (2019). doi:10.1063/1.5081094, URL http://aip.scitation.org/doi/10.1063/1.5081094

[23] D. Studer, *Probing atomic and nuclear structure properties of promethium by laser spectroscopy*, Ph.D. thesis, Johannes Gutenberg-Universität Mainz (2020). doi:10.25358/openscience-4863, URL https://openscience.ub.uni-mainz.de/handle/20.500.12030/4865

[24] F. Schwellnus, K. Blaum, C. Geppert, T. Gottwald, H.-J. Kluge, C. Mattolat, W. Nörtershäuser, K. Wies, K. Wendt, *The laser ion source and trap (LIST) – A highly selective ion source*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **266**(19-20), 4383–4386 (2008). doi:10.1016/j.nimb.2008.05.065, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X08007404

[25] D. A. Fink, T. E. Cocolios, A. N. Andreyev, S. Antalic, A. E. Barzakh, B. Bastin, D. V. Fedorov, V. N. Fedosseev, K. T. Flanagan, L. Ghys, A. Gottberg, M. Huyse, N. Imai, T. Kron, N. Lecesne, K. M. Lynch, B. A. Marsh, D. Pauwels, E. Rapisarda, S. D. Richter, R. E. Rossel, S. Rothe, M. D. Seliverstov, A. M. Sjödin, C. Van Beveren, P. Van Duppen, K. D. A. Wendt, *In-Source Laser Spectroscopy with the Laser Ion Source and Trap: First Direct Study of the Ground-State Properties of ¡math display="inline"¿ ¡mrow¿ ¡mmultiscripts¿ ¡mrow¿ ¡mi¿Po¡/mi¿ ¡/mrow¿ ¡mprescripts/¿ ¡none/¿ ¡mrow¿ ¡mn¿217¡/mn¿ ¡mo¿,¡/mo¿ ¡mn¿219¡/*, Physical Review X **5**(1), 011018 (2015). doi:10.1103/PhysRevX.5.011018, URL https://link.aps.org/doi/10.1103/PhysRevX.5.011018

[26] S. Raeder, H. Heggen, J. Lassen, F. Ames, D. Bishop, P. Bricault, P. Kunz, A. Mjøs, A. Teigelhöfer, *An ion guide laser ion source for isobar-suppressed rare isotope beams*,

Review of Scientific Instruments **85**(3), 033309 (2014). doi:10.1063/1.4868496, URL http://aip.scitation.org/doi/10.1063/1.4868496

[27] R. Catherall, W. Andreazza, M. Breitenfeldt, A. Dorsival, G. J. Focker, T. P. Gharsa, G. T J, J.-L. Grenard, F. Locci, P. Martins, S. Marzari, J. Schipper, A. Shornikov, T. Stora, *The ISOLDE facility*, Journal of Physics G: Nuclear and Particle Physics **44**(9), 094002 (2017). doi:10.1088/1361-6471/aa7eba, URL https://iopscience.iop.org/article/10.1088/1361-6471/aa7eba

[28] J. Montaño, T. Giles, A. Gottberg, *Design upgrade of the ISOLDE target unit for HIE-ISOLDE*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **317**, 430–433 (2013). doi:10.1016/j.nimb.2013.08.019, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X13008689

[29] D. R. Olander, V. Kruger, *Molecular Beam Sources Fabricated from Multichannel Arrays. III. The Exit Density Problem*, Journal of Applied Physics **41**(7), 2769–2776 (1970). doi:10.1063/1.1659313, URL http://aip.scitation.org/doi/10.1063/1.1659313

[30] R. Heinke, V. Fedosseev, T. Kieck, T. Kron, B. Marsh, S. Raeder, S. Richter, S. Rothe, K. Wendt, *Atom beam emersion from hot cavity laser ion sources*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **463**, 449–454 (2020). doi:10.1016/j.nimb.2019.04.026, URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X19302095

[31] R. Hasse, *No Title*, Bachelor thesis, Johannes Gutenberg-Universität Mainz (2021).

[32] F. Schneider, *First steps of the ECHo experiment: Penning-trap mass measurements of the 163Ho electron capture process and implantation of ultrapure Ho into microcalorimeter arrays*, Ph.D. thesis, Johannes Gutenberg-Universität Mainz (2016). doi:http://doi.org/10.25358/openscience-4067, URL https://openscience.ub.uni-mainz.de/handle/20.500.12030/4069

[33] L. Gamer, C. E. Düllmann, C. Enss, A. Fleischmann, L. Gastaldo, C. Hassel, S. Kempf, T. Kieck, K. Wendt, *Simulation and optimization of the implantation of holmium atoms into metallic magnetic microcalorimeters for neutrino mass determination experiments*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **854**, 139–148 (2017). doi:10.1016/j.nima.2017.02.056, URL https://linkinghub.elsevier.com/retrieve/pii/S0168900217302528

[34] C. Mattolat, S. Rothe, F. Schwellnus, T. Gottwald, S. Raeder, K. Wendt, T. Iguchi, K. Watanabe, *An All-Solid-State High Repetiton Rate Titanium:Sapphire Laser System For Resonance Ionization Laser Ion Sources*, in: AIP Conference Proceedings, AIP, 2009, pp. 114–119. doi:10.1063/1.3115586. URL http://aip.scitation.org/doi/abs/10.1063/1.3115586

[35] S. Rothe, B. A. Marsh, C. Mattolat, V. N. Fedosseev, K. Wendt, *A complementary laser system for ISOLDE RILIS*, Journal of Physics: Conference Series **312**(5), 052020 (2011). doi:10.1088/1742-6596/312/5/052020,
URL https://iopscience.iop.org/article/10.1088/1742-6596/312/5/052020

[36] C. Mattolat, *Spektroskopische Untersuchungen an Technetium und Silizium - Ein Festkörperlasersystem für die Resonanzionisationsspektroskopie*, Ph.D. thesis, Johannes Gutenberg-Universität Mainz (2010). doi:10.25358/openscience-1003,
URL https://openscience.ub.uni-mainz.de/handle/20.500.12030/1005

[37] A. Teigelhöfer, P. Bricault, O. Chachkova, M. Gillner, J. Lassen, J. P. Lavoie, R. Li, J. Meißner, W. Neu, K. D. A. Wendt, *Grating tuned Ti:Sa laser for in-source spectroscopy of Rydberg and autoionizing states*, Hyperfine Interactions **196**(1-3), 161–168 (2010). doi:10.1007/s10751-010-0171-x,
URL http://link.springer.com/10.1007/s10751-010-0171-x

[38] V. Sonnenschein, *Laser developments and high resolution resonance ionization spectroscopy of actinide elements*, Ph.D. thesis, University of Jyväskylä (2014).
URL http://urn.fi/URN:ISBN:978-951-39-6050-6

[39] V. Sonnenschein, I. D. Moore, S. Raeder, M. Reponen, H. Tomita, K. Wendt, *Characterization of a pulsed injection-locked Ti:sapphire laser and its application to high resolution resonance ionization spectroscopy of copper*, Laser Physics **27**(8), 085701 (2017). doi:10.1088/1555-6611/aa7834,
URL https://iopscience.iop.org/article/10.1088/1555-6611/aa7834

[40] T. Kalvas, O. Tarvainen, T. Ropponen, O. Steczkiewicz, J. Ärje, H. Clark, *IBSIMU : A three-dimensional simulation software for charged particle optics*, Review of Scientific Instruments **81**(2), 02B703 (2010). doi:10.1063/1.3258608,
URL http://aip.scitation.org/doi/10.1063/1.3258608

[41] T. Kalvas, *Development and use of computational tools for modelling negative hydrogen ion source extraction systems*, Ph.D. thesis, University of Jyväskylä (2013).
URL https://jyx.jyu.fi/bitstream/handle/123456789/42949/1/978-951-39-5421-5_vaitos05122013_PHYS.pdf

[42] T. Kalvas, *Ion Beam Simulator* (2015).
URL ibsimu.sourceforge.net

[43] M. Huyse, *Ionization in a hot cavity*, Nuclear Instruments and Methods in Physics Research **215**(1-2), 1–5 (1983). doi:10.1016/0167-5087(83)91284-X,
URL https://linkinghub.elsevier.com/retrieve/pii/016750878391284X

[44] V. Mishin, V. Fedoseyev, H.-J. Kluge, V. Letokhov, H. Ravn, F. Scheerer, Y. Shirakabe, S. Sundell, O. Tengblad, *Chemically selective laser ion-source for the CERN-ISOLDE on-line mass separator facility*, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **73**(4), 550–560 (1993). doi:10.1016/0168-583X(93)95839-W,
URL https://linkinghub.elsevier.com/retrieve/pii/0168583X9395839W

[45] J. Lawson, *Particle beams and plasmas*, Tech. rep. (1976).
URL https://cds.cern.ch/record/185933/files/CERN-76-09.pdf

[46] S. Richter, *Implementierung der Laserionenquellenfalle LIST bei ISOLDE und Validierung der Spezifikationen Effizienz und Selektivität*, Ph.D. thesis (2015). doi:http://doi.org/10.25358/openscience-1250,
URL https://openscience.ub.uni-mainz.de/handle/20.500.12030/1252

[47] M. Menat, *A quantitative study of scattering phenomena in an electromagnetic separator*, Canadian Journal of Physics **42**(1), 164–192 (1964). doi:10.1139/p64-013,
URL http://www.nrcresearchpress.com/doi/10.1139/p64-013

[48] M. Menat, G. Frieder, *Scattering phenomena in double-direction focusing electromagnetic mass separators*, Canadian Journal of Physics **43**(8), 1525–1542 (1965). doi:10.1139/p65-142,
URL http://www.nrcresearchpress.com/doi/10.1139/p65-142

[49] B. Marsh, *Instantaneous RILIS efficiency measurement method*, Tech. rep. (2020). doi:10.5281/zenodo.3931971.
URL https://zenodo.org/record/3931971

[50] Y. Liu, Q. Wu, X. Zhang, J. Liu, W. Huang, Y. Zhou, Z. Jia, Y. Zhai, L. Sun, *Design of a 180° electromagnetic isotope separator with inhomogeneous magnetic field*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **1006**, 165428 (2021). doi:10.1016/j.nima.2021.165428,
URL https://linkinghub.elsevier.com/retrieve/pii/S0168900221004125

[51] H. Mirzaeian, S. Manjooran, A. Major, *A simple technique for accurate characterization of thermal lens in solid state lasers*, 2014, p. 928802. doi:10.1117/12.2075117.
URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2075117

[52] A. Siegman, *Lasers*, University science books, 1986.

[53] R. Horn, *Aufbau eines Systems gepulster, abstimmbarer Festkörperlaser zum Einsatz in der Resonanzionisations-Massenspektroskopie*, Phd thesis, Johannes Gutenberg-Universität Mainz (2003).

[54] H. L. Fang, R. L. Swofford, *Analysis of the thermal lensing effect for an optically thick sample—A revised model*, Journal of Applied Physics **50**(11), 6609–6615 (1979). doi:10.1063/1.325911,
URL http://aip.scitation.org/doi/10.1063/1.325911