Case Studies in Many-Body Physics

Ana Samolov
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/physics_etds

Part of the Atomic, Molecular and Optical Physics Commons, and the Plasma and Beam Physics Commons

Recommended Citation
Samolov, Ana. "Case Studies in Many-Body Physics" (2012). Doctor of Philosophy (PhD), Dissertation, Physics, Old Dominion University, DOI: 10.25777/maaq-cm05
https://digitalcommons.odu.edu/physics_etds/83

This Dissertation is brought to you for free and open access by the Physics at ODU Digital Commons. It has been accepted for inclusion in Physics Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.
CASE STUDIES IN MANY-BODY PHYSICS

by

Ana Samolov
BSc., December 2005, Belgrade University
MS, May 2008, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of
DOCTOR OF PHILOSOPHY
PHYSICS

OLD DOMINION UNIVERSITY
May 2012

Approved by:

Alexander Gedunov (Director)

Ian Balitsky (Member)

Geoff Krafft (Member)

Svetozar Popović (Member)

Ali Beskok (Member)
The many-body problem refers to any physical problem made of more than two interacting particles. With increasing number of particles in a system, their coupling and entanglement becomes more complex, and there is no general analytic solution even for a three-body classical or quantum systems. However, some of the most fascinating phenomena in nature are products of collective effects. Therefore, significant efforts have been made in both experiment and theory to unravel some specific many-body problems. If we look at still unanswered physics questions we see that for most of these problems addressing the many-body interactions is a key issue. This field of research is very active, and with the theory relying on multiple approximations specific for the problem at hand, it has become one of the most computationally intensive areas of physics. In this work we address several many-body problems that are still puzzling the scientific community, using different theoretical and computational techniques:

1. Recent experiments in atomic physics considering the proton impact ionization of hydrogen revealed that experimental observations cannot be explained with the available theoretical models, developed for more complex helium atom. We used the approximate solution for a three-body Coulomb system to calculate double differential cross sections for proton impact ionization of hydrogen atom, to describe the new experimental findings.

2. One of the central problems in the accelerator science is the interaction of a charged particle beam within itself and matter. Thus, it is crucial that we understand the collective effects governing the scattering of many particles in the bunch on multi center targets. We have developed the particle-particle computational code, based on classical scattering theory, which allows us to include close range interactions between the particles in the study of these many-body effects.
3. In this work we have also considered plasmas, which are manifestation of many-body collective effects. To study the formation of plasmoid-like object in supersonic flow microwave discharge, we have refined the tomographic diagnostic method, so we can take a glance inside this plasma object without disturbing it. The tomographic analysis provided us with spatial distributions of plasma constituents that we need for understanding of the collective-effects in its formation.
Copyright, 2012, by Ana Samolov, All Rights Reserved.
ACKNOWLEDGMENTS

I would like to start by thanking my advisor, Dr. Alex Godunov, for his patience and support throughout my research. His enthusiasm and knowledge about physics were the most inspiring. Additionally, I would like to thank Dr. Leposava Vušković and Dr. Svetozar Popović for their invaluable input about the problems that shaped this thesis. Also, I have to thank all three of them for their immense friendship and care that made the very nurturing environment for me to grow both as a physicist and as a person.

I would like to express my appreciation for my graduate committee whose invaluable input and comments has stirred my research work over the years. To the staff, faculty and graduate students at the department of physics at ODU, thank you for your support, encouragement and all the laughs for the last six years. I especially want to thank my class mates and my lab mates Janardan Upadhyay and Milka Nikolić, whose depth of knowledge never ceases to amaze me, for making the work in the lab so much fun. To my flat mates Marija, Mladen, Miloš and Ivan thank you for all the fun and loving memories that I will cherish for always.

To my mom, my dad and my sister and my life long very best friends in Belgrade, I want to especially thank you for giving me the courage to embark on this amazing journey and for supporting me through out the whole experience, I truly would not have made it without you.

Most importantly I want to thank my Filip for sticking with me through thick and thin. His kindness, intellect and love were most uprising making everything possible. "I am forever yours, faithfully".
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>ix</th>
</tr>
</thead>
</table>

## Chapter

1. **INTRODUCTION** 1

2. **3 - BODY PROBLEM: IONIZATION** 4
   2.1 *THE "PURE" THREE-BODY SYSTEM* 4
   2.2 *COLLISIONAL THEORY* 5
   2.3 *THE MODEL* 6
   2.4 *RESULTS AND DISCUSSION* 17

3. **MANY-BODY PROBLEM: SIMULTANEOUS SCATTERING** 26
   3.1 *SIMULTANEOUS VS. MULTIPLE SCATTERING* 27
   3.2 *CLASSICAL MODEL AND SIMULATION* 28
   3.3 *SIMULTANEOUS SCATTERING EFFECTS* 34

4. **MANY-BODY PROBLEM IN PLASMA: TOMOGRAPHY** 51
   4.1 *PLASMOID* 53
   4.2 *EXPERIMENT* 53
   4.3 *TOMOGRAPHY* 56
   4.4 *INVERSION METHODS* 62
   4.5 *TOMOGRAPHIC ANALYSIS OF EXPERIMENTAL RESULTS* 71

5. **CONCLUSION** 76

## BIBLIOGRAPHY

## APPENDICES

A. **BEAR II** 83
B. **SESAME** 85
C. **QUANTILES AND Q-Q PLOTS** 109
D. **DIRECT INTEGRATION CODE** 111
E. **FILTERED BACK PROJECTION CODE** 129

## VITA 149
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Schematic representation of the collisional system for the process of proton impact ionization of atomic hydrogen.</td>
</tr>
<tr>
<td>2.</td>
<td>Experimental setup on single ionization measuring the momentum vectors of the scattered projectile and recoil ion.</td>
</tr>
<tr>
<td>3.</td>
<td>Double differential cross section for fixed energy loss of $\Delta E = 30$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).</td>
</tr>
<tr>
<td>4.</td>
<td>Double differential cross section for fixed energy loss of $\Delta E = 40$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).</td>
</tr>
<tr>
<td>5.</td>
<td>Double differential cross section for fixed energy loss of $\Delta E = 50$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).</td>
</tr>
<tr>
<td>6.</td>
<td>Double differential cross section for fixed energy loss of $\Delta E = 53$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).</td>
</tr>
<tr>
<td>7.</td>
<td>Scattering schematics of a positively charged ion colliding with atomic hydrogen.</td>
</tr>
<tr>
<td>8.</td>
<td>SESAME scheme of operation; In blue - target neutrals, in red - target ions, in black - projectiles.</td>
</tr>
</tbody>
</table>
9. The change of the direction of motion of a particle due to the change of its momentum's transverse component. ............................................. 35

10. The simultaneous scattering effect for the target density of $\rho = 10^{22} \text{ cm}^{-3}$. ................................................................. 36

11. The q-q plot for simultaneous scattering effect for the target density of $\rho = 10^{22} \text{ cm}^{-3}$. ................................................................. 36

12. The simultaneous scattering effect for the target density of $\rho = 10^{24} \text{ cm}^{-3}$. ................................................................. 37

13. The q-q plot for simultaneous scattering effect for the target density of $\rho = 10^{24} \text{ cm}^{-3}$. ................................................................. 37

14. The simultaneous scattering effect for the target density of $\rho = 10^{26} \text{ cm}^{-3}$. ................................................................. 38

15. The q-q plot for simultaneous scattering effect for the target density of $\rho = 10^{26} \text{ cm}^{-3}$. ................................................................. 38

16. The simultaneous scattering effect for the 10% ionized target. ........ 39

17. The q-q plot for simultaneous scattering effect for the 10% ionized target. 40

18. The simultaneous scattering effect for the 100% ionized target. ........ 40

19. The q-q plot for simultaneous scattering effect for the 100% ionized target. 41

20. The simultaneous scattering effect for the 8 MeV $\mu^+$ .................. 42

21. The q-q plot for simultaneous scattering effect for the 8 MeV $\mu^+$ . 42

22. The change in direction of particle motion due to the change of its momentum's longitudinal component. ................................. 43

23. The simultaneous scattering effect for the 7.5 MeV $p^+$ .................. 44

24. The q-q plot for simultaneous scattering effect for the 7.5 MeV $p^+$ . 44

25. The simultaneous scattering effect for the 750 MeV $p^+$ .................. 45

26. The q-q plot for simultaneous scattering effect for the 750 MeV $p^+$ . 45

27. The simultaneous scattering effects dependence on the momentum transfer due to the recoil of target atoms. ................................. 46
28. The simultaneous scattering effects dependence on the momentum transfer due to the recoil of target atoms. ........................................... 47
29. The simultaneous scattering effects compared for single particle and bunch tracking, 75 keV $p^+$ ............................................................... 48
30. Evolution of the 15 MeV $\mu^+$ bunch. Different colored dots correspond to spatial distributions of the same bunch at different times. ............... 50
31. Scheme of supersonic flow experimental setup. .......................... 54
32. Automated measurement system and plasmoid. .......................... 55
33. The tomographic projection $g(p, \theta)$ is the collection of parallel ray integrals through the object $f$ in the direction specified by $\theta$; the Radon transform is the set of all such angular projections for $\theta \in [0, \pi]$ 57
34. The two-dimensional Gaussian distribution function $f(x, y) = \exp(-x^2 - y^2)$. ................................................................. 59
35. The Radon transform of Gaussian distribution function 
$\mathcal{R}\{\exp(-x^2 - y^2)\} = \sqrt{\pi} \exp(-p^2)$ ........................................ 60
36. The Inverse Radon transform of Gaussian distribution function 
$\mathcal{R}^{-1}\{\exp(-x^2 - y^2)\} = \sqrt{\pi} \exp(-p^2)$ using the direct integration method. ...................................................... 65
37. Ram-Lak and Shepp-Logan filters in both frequency and spatial domain. 67
38. The Inverse Radon transform of Gaussian distribution function 
$\mathcal{R}^{-1}\{\exp(-x^2 - y^2)\} = \sqrt{\pi} \exp(-p^2)$ using the Filtered Back Projection with Ram-Lak filter. ............................................ 70
39. Population of the Ar I $[3s^23p^5(2P_{1/2})^4p \rightarrow 3s^23p^5(2P_{3/2})^4s]$ at 706.72 nm, 2.4 Torr, reconstructed. ........................................... 73
40. Population of the Ar I $[3s^23p^5(2P_{1/2})^4p \rightarrow 3s^23p^5(2P_{3/2})^4s]$ at 714.70 nm, 2.4 Torr, reconstructed. ........................................... 74
41. Pseudo code for BEAR II. .................................................. 84
CHAPTER 1

INTRODUCTION

The most fascinating phenomena in nature are manifestations of collective effects. The collective behavior is always related to the so-called many-body problem. The many-body problem encompasses all physical problems that consist of more than two interacting particles. When a number of particles in a system increases, their coupling and entanglement becomes more and more complex, and there is no general analytic solution even for the simplest three-body classical or quantum systems. If we look at still unanswered physics questions, we see that a key issue for most of these problems is addressing the many-body interactions. The field of many-body physics is most intriguing, because without a general solution, every problem needs to be addressed in a specific way. In that sense many-body physics is a very wide and a very active research field driven by vast applications. With the theory relying on multiple approximations specific for the problem at hand, this field is becoming one of the most computationally intensive areas of research. This work is a compilation of several many-body problems that are still puzzling the scientific community. We approach them using different theoretical and computational techniques from quantum mechanical, classical to semi-empirical.

With Rutherford's scattering experiment, collisional experiments became the probing tool for studying the structure of matter and have also provided rich information about many particle dynamics. Atomic collision studies had a central place in testing the theoretical models of few-body interactions, for several reasons. First of all, the fundamental force in the domain of atomic physics is the electromagnetic force, which is completely understood. Then, any divergence of the theory from experimental data may be associated with the few-body aspects of the theoretical model. Also, the state-of-the-art atomic collision experiments support the study of systems with relatively small number of particles. This way the unknown features of few-body correlations would not be lost in the statistics of huge number of particles. Recent experiments in atomic physics on the proton impact ionization of hydrogen have revealed that experimental observations can not be explained with available theoretical models. The process of proton impact ionization of atomic hydrogen is
particularly suited for the study of three-body dynamics. It has three unbound particles interacting in the exit channel and is free of any complications introduced by correlation of target electrons or exchange processes. We used the approximate solution for a three-body Coulomb system derived from first principles for calculation of double differential cross sections for the ionization of hydrogen by proton impact. The differential cross sections are the ones carrying the most information on collisional dynamics, therefore they were our main focus in order to describe the new experimental findings, and better understand the 3-body dynamics.

An additional problem that attracted our attention is one of the central problems in the accelerator science. It considers the interaction of a charged particle beam within itself and matter. Thus, the understanding of the collective effects governing the scattering of many particles in the bunch on multiple target centers is crucial.

This area of research was initiated by the discovery of highly energetic particle emissions from the radioactive material. Soon afterwards, it became obvious that these charged particles can penetrate the matter, and that their interactions with the surrounding matter are dictated by the collisions with the target atoms. Due to this combined effect of multiple scattering, particles are experiencing angular and energy straggling. There has been significant activity during the last century in both theory and experiment to describe these collective effects. Traditionally, multiple scattering was treated as a number of successive binary collisions that are statistically independent form each other. The effect from the other particles may or may not have been included through different approximations. Historically multiple scattering theory has treated the target and projectile aspects of the problem separately. In our classical approach to this problem we are considering what we call simultaneous scattering, which refers to scattering of charged particles off of the total potential of the target atoms and all the other particles in the bunch at all times. For this purpose we have developed a particle-particle computational code, based on classical scattering theory, that accounts for close range interactions between the particles that are needed for studying these many-body effects.

In this work, alongside the quantum and classical we have also used a semi-empirical approach to the many-body problem. We have studied plasma, an object which is known to be a manifestation of collective effects. Plasma is characterized by collective effects and entanglement that far exceed the ones observed in liquids or solids.
The information obtained from plasma experiments are all integrated over the whole plasma volume. Though these are necessary for evaluating macroscopic plasma parameters that are important for practical application, they do not tell us much about internal dynamics of the observed plasma object. For that purpose we propose the use of plasma emission tomography which provides a way of transforming the integral data into the spatial population distributions of the plasma constituents. We use it as a magnifying glass to look inside the plasma object without disturbing it.

To understand the collective-effects in governing the formation of a plasmoid-like object in supersonic flow microwave discharge in pure argon, observed at the Atomic Beam Lab at the Department of Physics, Old Dominion University, we have refined several tomographic reconstruction methods. The tomographic analysis was done on emission spectroscopy data for the reconstruction of plasmoid spatial population distribution, needed for studying collective-effects responsible for its formation.

The rest of this thesis is organized as follows. In Chapter 2, we present our quantum mechanical three-body model used for description of few-body dynamics in proton impact ionization of hydrogen. In Chapter 3 we present our classical model and computational code developed for the treatment of the problem of simultaneous scattering of a beam of charged particles while interacting with a material target. Chapter 4. is focused on the discussion of plasma tomography as a diagnostic technique for studying collective effects in plasmas and its use for the characterization of plasmoid, observed in supersonic microwave flow. Finally, Chapter 5 presents the summary with concluding remarks of this research.
CHAPTER 2

3 - BODY PROBLEM: IONIZATION

Collisional experiments, since Rutherford’s pioneering work, have been probing the structure of matter and providing rich information on mechanisms of many particle dynamical processes. Atomic collision studies, moreover, directly address the fundamentally important and still unsolved many-body problem. There are a couple of reasons that make atomic collision experiments particularly suitable for testing the theoretical treatment of few-body problem. The first one is that the underlying force on atomic scale is the electromagnetic force, which is completely understood. Therefore any discrepancies between theory and experiment are attributed to the few-body aspects of the model. The second reason is that advanced atomic collision experiments allow us to study systems with relatively small number of particles. Hence, any lack of understanding of the few-body phenomenology would not be masked by the statistics of huge number of particles. The novel kinematically complete atomic collision experiments serve as test beds for theoretical models on an individual particle level.

2.1 THE "PURE" THREE-BODY SYSTEM

Ionization processes are particularly insightful when studying few-body problems. The single atom impact ionization processes have at least three unbound particles in their exit reaction channel. Out of these processes the ones involving atomic hydrogen are deemed as three-body systems, because they are not affected by complications resulting from electron correlation in many-electron targets. The process of proton impact ionization of atomic hydrogen, furthermore, presents a "pure" three-body system since it has exactly three unbound particles in the final state, and since the proton is a projectile; the theory is not concerned with indistinguishable particles and complicated many-electron states. The proton impact ionization is also interesting from the application point of view. It accounts for large energy loss of the fast proton in materials, so research fields such as radiation damage, radiation biology, fusion science, and plasma physics would all benefit from proton impact ionization cross
section data. Another thing is that lately the experimental measurements of double differential cross sections for this system became available. Experimental physicists only recently managed to overcome the challenging task of producing atomic hydrogen from the dissociation process of $H_2$, cool enough to be able to do the kinimatically complete measurements [1]. It has turned out that conventional theoretical methods had some trouble reproducing the experimental data. Thus, we start our study of the many-body problem with the case of proton impact ionization of atomic hydrogen, $p^+ + H \rightarrow p^+ + H^+ + e^-$, both for its relevance to the other areas of science and its three-body aspects.

2.2 COLLISIONAL THEORY

From the theoretical point of view even the simplest three-body breakup processes pose a serious challenge for quantum theory. The infinite number of decay channels and an infinite reaction space make it very difficult to determine time-independent boundary conditions. Additionally, the long range character of Coulomb interaction constrains the motion of the particles even at macroscopic distances. Therefore, calculations of ionization cross sections have to combine detailed modeling of collisional dynamics and accurate computation of structural properties of the target atom.

2.2.1 CROSS SECTIONS

For practical applications, the total cross sections and collisional rates are most important. However, the total cross sections are not particularly suited for testing theoretical description of few-body dynamics in ionizing collisions. In the integration over the kinematic parameters of the particles in the exit channel, much information about collision mechanisms and atomic structure is lost. For fundamental research, therefore, the differential cross sections for ionization are of primary interest. In principle, the more differential the cross section, the more we can learn from it about the driving mechanisms of the process. The fully differential cross sections provide a wealth of information about single electron ionization processes, because the two vector momenta out of three are registered in coincidence, and the momentum of the third particle is reconstructed from the laws of conservation of energy and momentum. Complete experiments in the physics of proton impact ionization of atomic hydrogen, due to complex technical problems have only recently been initiated [1]. The measurements of doubly differential cross sections (DDCS) as a function of
the angle and energy of the particles are more common, and as such DDCS will be the subject of our following discussions. We take particular interest in DDCS as a function the projectile scattering angle and fixed energy losses, since they proved to be a subject of strong interest for the study of projectile-target nucleus interaction, which is learned to be from both theoretical and experimental analysis a very important collisional mechanism [2, 3].

2.2.2 THEORETICAL METHODS

Before proceeding with the description of the theoretical model that we used for the calculation of DDCS for the process of proton impact ionization of atomic hydrogen, we will give a brief overview of the available theoretical methods which are suitable for ion-atom collisions.

The Born approximation is maybe the most basic of all approaches in computing the scattering amplitudes. The Born Approximation is just the first couple of terms in the infinite Born series which converges fast in the case if the interaction potential is weak enough, or the collisional energies are high. Therefore, our basic approach is useful when the potential is weak enough for all the higher terms in the Born Series to be neglected. The question is what to do when the potential is too strong to rely on the Born Approximation. In these cases the alternative is the distorted-wave Born approximation (DWBA). The DWBA is applicable whenever the interaction potential can be written as the sum of two $V_a = V_a^I + V_a^{II}$, where the amplitude of the first potential is exactly known, and the second potential is just a small perturbation. Here the role of the first potential is to scatter the projectile and to distort the waves seen by the $V_a^{II}$. Considering this, it is understood that one can treat the second term as the Born approximation for scattering by $V_a^{II}$ in the presence of $V_a^I$.

The appealing property of the DWBA is the fact that it is usually possible and convenient to choose $V_a^I$ so that the scattering amplitude for the first potential vanishes and what is left is the first Born Approximation of the distorted waves scattered on the potential $V_a^{II}$. Also, one can use for the $V_a^I$ Coulomb interaction for which the scattering waves are exactly known.

The DWBA over the years has evolved in a whole class of methods, that includes Coulomb Distorted Wave (CDW), CDW with Eikonal Initial State (CDW-EIS), etc. Combined with the three-body boundary conditions their importance is growing in the study of the many body dynamics, or dynamic correlations.
2.3 THE MODEL

Here we will present a consistent analysis of DDCS for proton impact ionization of atomic hydrogen. We consider the role of the Coulomb interaction in the final state between the scattered proton, the ejected electron, and the recoil hydrogen ion, so-called post collisional interaction (PCI), as well as the two-step transition that includes the interaction of the projectile with the target nucleus (PT). We aim to account for all the principle mechanisms in our model and to be specific in our analysis to be able to separate their contributions. Our calculations are based on two different models of the collisional dynamics. The first model is the approximation solution of Faddeev-Merkuriev equations for the three-body Coulomb problem. The feasibility of this method has already been demonstrated for both the description of DDCS of direct ionization of helium [4] and auto-ionizing resonances of helium in electron emission spectra excited by the fast ion impact [5]. The second model is the expansion of the transition amplitude in the Born series over the projectile-target interaction up to second order. Our results for the DDCS of single ionization of atomic hydrogen by 75 keV proton impact will be presented in comparison with experimental results [1] and available theoretical calculations. We use atomic units throughout our work.

2.3.1 THEORY

The double differential cross section for single ionization as a function of the scattered projectile solid angle $\Omega_p$ and the energy of the scattered projectile $E_p$ is defined as:

$$\frac{d^2\sigma}{d\Omega_p dE_p} = \frac{K_f k_e}{K_i} \int d\Omega_e |f_{\text{dir}}|^2$$

where $K_i$ and $K_f$ are the momenta of the incoming and outgoing projectile, $k_e$ is the momentum of the ejected electron, $\Omega_e$ is the solid angle element in the direction of ejected electron.

The scattering amplitude $|f_{\text{dir}}|$ is given by:

$$f_{\text{dir}} = - (2\pi)^2 \mu \langle \Psi_f^- | V_i | \Psi_i \rangle$$

with
\[ \Psi_i = \frac{1}{(2\pi)^{3/2}} \varphi_i(\vec{r}_{23}) \exp(i\vec{K}_i\vec{R}) \] 

the wavefunction of the unperturbed initial state and \( V_i \) is the interaction potential between the projectile of charge \( Z_p \) and the atom with the nuclear charge \( Z_t \), namely

\[ V_i = \frac{Z_p Z_t}{R} - \frac{Z_p}{r_{12}} \]

where \( \mu \) is the reduced mass of the projectile and the target, \( \varphi_i(\vec{r}_{23}) \) is the wavefunction of the hydrogen atom in the ground state, \( \vec{r}_{ij} \) is the relative coordinate of the particles \( i \) and \( j \), \( \vec{R} \) is the position of the projectile relative to the target.

FIG. 1: Schematic representation of the collisional system for the process of proton impact ionization of atomic hydrogen.
Subscripts 1-3 correspond to the projectile, the ejected electron and the recoil ion respectively. $\Psi_f^{(-)}$ is the exact wavefunction of the final state for the Coulomb system "projectile + Hydrogen atom" with Hamiltonian

$$H = -\frac{1}{2\mu} \nabla^2_k + V_t + \frac{1}{2} \nabla^2_{r_{23}} - \frac{Z_t}{r_{23}}.$$  \hfill (5)

The total energy of the system is

$$E = \frac{1}{2\mu} K_f^2 + \frac{1}{2} k_e^2.$$  \hfill (6)

### 2.3.2 THREE-BODY MODEL

An approach to three-body problem derived from the first principles was suggested by Ludvig Dmitrievich Faddeev [6]. He was the first to consider the mathematical aspects of the scattering theory for a system of three particles interacting through short-range potentials. However, the Faddeev equations are not applicable to the Coulomb scattering problem, due to the long range character of the Coulomb interaction. A new form of Faddeev equations for the three-body Coulomb problem was developed by Merkuriev [7] providing us with a rather interesting and promising approach for theoretical treatment of impact ionization. Nevertheless, this is an extraordinary challenging task, even in the asymptotic region [8, 9]. The direct numerical integration of the fundamental equations with correct asymptotic Coulomb behavior for all regions of the configuration space for the three-body Coulomb system is computationally very intensive, and so far the practical calculations of this kind were not performed. However, there have been number of attempts to use analytical or semi-analytical form of an approximate wavefunction for three-body Coulomb system for practical calculations [10-16] in atomic collisions. Godunov et al. have successfully applied the approximate solution of the Faddeev-Merkuriev equations to the proton impact ionization of helium, and we start our derivation from the same wavefunction used in their work [4]. The approximate final-state wavefunction for ionization of hydrogen by a projectile of charge $Z_p$ can be written as [4]

$$\Psi_f^{(-)} \approx \frac{1}{(2\pi)^{3/2}} \psi_{k_{23}}^{(-)}(\vec{r}_{23}) \exp(iK_f\vec{R}) \Phi_{k_{12}}^{(-)}(\vec{\nu}_{12}, \vec{r}_{12}) \Phi_{k_{13}}^{(-)}(\vec{\nu}_{13}, \vec{R})$$  \hfill (7)

where $\psi_{k_{23}}^{(-)}(\vec{r}_{23})$ is the continuum wavefunction of the ejected electron in the field of recoil ion and
\[ \Phi_{k_{ij}}^{(-)}(\nu_{ij}, \vec{r}_{ij}) = f_{c}^{(-)}(\nu_{ij})_1 F_1[i\nu_{ij}, 1, -i(k_{ij}r_{ij} + \vec{k}_{ij}\vec{r}_{ij})] \] (8)

is a Coulomb distortion factor. Here

\[ f_{c}^{(-)}(\nu_{ij}) = \exp\left(-\frac{\pi \nu_{ij}}{2}\right)\Gamma(1 - \nu_{ij}) \quad \nu_{ij} = \frac{Z_{i}Z_{j}m_{ij}}{k_{ij}} \] (9)

with the \( \vec{k}_{ij} \) the momenta of the particles \( i \) and \( j \), \( m_{ij} \) their reduced mass, \( Z_{i} \) is the charge of the particle \( i \), \( \Gamma \) is the gamma function and \( 1 F_1 \) the confluent hypergeometric function.

The approximate final-state wavefunction (7) is valid when all three particles are well separated, for the other cases it may be improved by using dynamical screening for Sommerfeld parameters \( \nu_{ij} \). For the collisional velocities that we will consider here, the approximate final-state wavefunction (7) is still feasible \[4\].

The numerical evaluation of the scattering amplitude (2) with the wavefunction (7) is possible using multi-dimensional integration \[17\]. Practically, it is very computationally intensive but the computational efforts may be reduced by utilizing the properties of the Fourier transform of the Coulomb functions. It is very useful to work in the momentum space when evaluating the ionization amplitude \( f_{dir} \).

Using Bethe’s integral, which expresses the Coulomb potential in momentum space,

\[ \frac{1}{R} = \frac{1}{2\pi^2} \int \frac{ds}{s^2} \exp(-isR) \] (10)

we can rewrite our interaction interaction (4),

\[ V_{i} = \frac{Z_{p}}{2\pi^2} \int \frac{ds}{s^2} \exp(-isR)[Z_{i} - \exp(-isR_{12})]. \] (11)

The Fourier transform of the Coulomb distortion factor (8), is given by

\[ \Phi_{k_{ij}}^{\pm}(\nu_{ij}, \vec{p}) = \int d\vec{r}_{ij} \exp(i\vec{p}\cdot\vec{r}_{ij})\Phi_{k_{ij}}^{\pm}(\nu_{ij}, \vec{r}_{ij}) \] (12)

\[ \Phi_{k_{ij}}^{\pm}(\nu_{ij}, \vec{p}) = \frac{1}{(2\pi)^3} \int d\vec{p} \exp(-i\vec{p}\cdot\vec{r}_{ij})\Phi_{k_{ij}}^{\pm}(\nu_{ij}, \vec{p}). \] (13)

Substituting the momentum form of the interaction potential and Fourier representation of the Coulomb distortion factor \( \Phi_{k_{ij}}^{\pm}(\nu_{ij}, \vec{r}_{ij}) \) into the transition amplitude.
\[ f_{\text{dir}} \] and ignoring correction of the order of \( m_e/m_p \), where \( m_e \) is the electron mass and \( m_p \) is the projectile mass, yields

\[
f_{\text{dir}} = -\frac{2\mu Z_p}{(2\pi)^6} \int dp \tilde{\Phi}_{k;13}^{+}(\nu_{12}, \vec{p}) \int \frac{d\vec{s}}{s^2} \tilde{\Phi}_{k;13}^{+}(\nu_{13}, \vec{Q} - \vec{p} - \vec{s}) [Z_t T_{f1}(\vec{p}) - T_{f1}(\vec{p} + \vec{s})] \tag{14}
\]

where \( \vec{Q} = \vec{K}_i - \vec{K}_f \) is the momentum transfer. The atomic form factors \( T_{fi} \) are defined as

\[
T_{f1}(\vec{p}) = \int d\vec{r}_{23} \Psi_{k;23}^{(-)*}(\vec{r}_{23}) \exp(i\vec{p} \cdot \vec{r}_{23}) \varphi_i(\vec{r}_{23}). \tag{15}
\]

It is useful to explore separately the roles of the interaction of the projectile with active electron and the target nucleus

\[
f_{\text{dir}} = f_{\text{pe}} + f_{\text{pt}} \tag{16}
\]

with the corresponding form factors \( T_{f1}(\vec{p} + \vec{s}) \) and \( T_{f1}(\vec{p}) \).

Introducing the new variable \( \vec{k} = \vec{Q} - \vec{p} - \vec{s} \) one obtains the ionization amplitude responsible for scattering the projectile by the active electron

\[
f_{\text{pe}} = \frac{2\mu Z_p}{(2\pi)^6} \int d\vec{k} T_{f1}(\vec{Q} - \vec{k}) \tilde{\Phi}_{k;13}^{+}(\nu_{13}, \vec{k}) \int \frac{d\vec{s}}{s^2} \tilde{\Phi}_{k;13}^{+}(\nu_{12}, \vec{Q} - \vec{k} - \vec{s}) \tag{17}
\]

To reduce the integration over \( \vec{k} \) we use the following identity:

\[
T_{f1}(\vec{Q} - \vec{k}) = T_{f1}(\vec{Q}) + [T_{f1}(\vec{Q} - \vec{k}) - T_{f1}(\vec{Q})]. \tag{18}
\]

Hence, the amplitude \( f_{\text{pe}} \) takes the form,

\[
f_{\text{pe}} = f_{\text{pe}}^{c1} + f_{\text{pe}}^{c2}. \tag{19}
\]

Using the Fourier convolution theorem one has:

\[
\frac{1}{(2\pi)^2} \int d\vec{s} \tilde{\Phi}_{k;ij}^{+}(\nu_{ij}, \vec{Q} - \vec{k} - \vec{s}) = \int \frac{d\vec{R}}{\vec{R}} \Phi_{k;ij}^{+}(\nu_{ij}, \vec{R}) \exp(i(\vec{Q} - \vec{k}) \cdot \vec{R}). \tag{20}
\]

Then, it is straightforward to show that

\[
f_{\text{pe}}^{c1} = \frac{\mu Z_p}{(2\pi)^4} T_{f1}(\vec{Q}) \int d\vec{k} \tilde{\Phi}_{k;13}^{+}(\nu_{13}, \vec{k}) \int \frac{d\vec{R}}{\vec{R}} \tilde{\Phi}_{k;12}^{+}(\nu_{12}, \vec{R}) \exp(i(\vec{Q} - \vec{k}) \cdot \vec{R}) \tag{21}
\]

\[= \frac{\mu Z_p}{2\pi} T_{f1}(\vec{Q}) \int \frac{d\vec{R}}{\vec{R}} \tilde{\Phi}_{k;12}^{+}(\nu_{12}, \vec{R}) \tilde{\Phi}_{k;13}^{+}(\nu_{13}, \vec{R}) \exp(i(\vec{Q} - \vec{R}) \cdot \vec{R}) \tag{22}
\]
The integration over $R$ in equation (22) can be carried out using the Nordsieck's technique (1954)

$$
\int \frac{d^3r}{r} \exp(-\alpha r + i\vec{p} \cdot \vec{r}) \, \frac{1}{F_1(i\nu_1, 1, i(k_1 r + \vec{k}_1 \cdot \vec{r})) \, F_1(i\nu_2, 1, i(k_2 r + \vec{k}_2 \cdot \vec{r}))}
= \frac{4\pi}{\alpha^2 + p^2} \left( 1 + \frac{2\vec{p} \cdot \vec{k}_1 - 2i\alpha k_1}{\alpha^2 + p^2} \right)^{-\nu_1} \left( 1 + \frac{2\vec{p} \cdot \vec{k}_2 - 2i\alpha k_2}{\alpha^2 + p^2} \right)^{-\nu_2} F_1(\nu_1, \nu_2, 1, Z)
$$

(23)

where

$$
Z = 2 \left( \frac{2(\vec{p} \cdot \vec{k}_1 - i\alpha k_1)(\vec{p} \cdot \vec{k}_2 - i\alpha k_2) + (\alpha^2 + p^2)(k_1 k_2 - \vec{k}_1 \cdot \vec{k}_2)}{(\alpha^2 + p^2 + 2\vec{p} \cdot \vec{k}_1 - 2i\alpha k_1)(\alpha^2 + p^2 + 2\vec{p} \cdot \vec{k}_2 - 2i\alpha k_2)} \right).
$$

(24)

This gives us the amplitude of ionization $f_{pc}^{cl}$

$$
f_{pc}^{cl} = 2\mu Z_p T_{fi}(Q) f_c^+(\nu_{12}) f_c^+(\nu_{13})(1 + d_{12})^{\nu_{12}}(1 + d_{13})^{\nu_{13}} F_1(-i\nu_{12}, -i\nu_{13}, 1, X)
$$

(26)

where $d_{ij}$ and $X$ are defined as:

$$
d_{ij} = \frac{2\vec{k}_{ij} \cdot \vec{Q}}{Q^2}, \quad X = 2 \frac{(\vec{k}_{12} \cdot \vec{Q})(\vec{k}_{13} \cdot \vec{Q}) - Q^2(\vec{k}_{12} \cdot \vec{k}_{13} - k_{12} k_{13})}{(Q^2 + 2\vec{k}_{12} \cdot \vec{Q})(Q^2 + 2\vec{k}_{13} \cdot \vec{Q})}.
$$

(27)

The derived amplitude for ionization $f_{pc}^{cl}$ (26) can also be written in the form [11,18]

$$
f_{pc}^{cl} = K_{dir} f_{dir}^{bl}(Q)
$$

(28)

where $f_{dir}^{bl}$ is the amplitude for direct ionization in the first Born approximation

$$
f_{dir}^{bl}(Q) = -\frac{\mu}{2\pi} \langle \Psi_{k_{23}}^{(-)} \exp(i\vec{K}_{fi} \cdot \vec{R}) | V_i | \varphi \exp(i\vec{K}_{fi} \cdot \vec{R}) \rangle = \frac{2\pi Z_p T_{fi}(Q)}{Q^2}
$$

(29)

and $K_{dir}$ is a factor accounting for the Coulomb interaction in the final state with the explicit form of:

$$
K_{dir} = f_c^+(\nu_{12}) f_c^+(\nu_{13})(1 + d_{12})^{\nu_{12}}(1 + d_{13})^{\nu_{13}} F_1(-i\nu_{12}, -i\nu_{13}, 1, X).
$$

(30)

The same expression (28) for $f_{pc}^{cl}$ amplitude can be derived from the equation (2) using the so-called peaking approximation [10,19]. Within this approximation, the $V_{pe}$
interaction is the only one providing a non-zero contribution. However, in the classical limit, projectile scattering by active electrons is restricted to a maximum angle of 0.55 mrad, meaning that the peaking amplitude is not valid for larger scattering angles.

For the amplitude $f_{pe}^{c2}$ we have:

$$f_{pe}^{c2} = \frac{2\mu Z_p^2}{(2\pi)^6} \int d\bar{k} |T_{\bar{f}_i}(\bar{Q} - \bar{k}) - T_{\bar{f}_i}(\bar{Q})| \tilde{\Phi}_{k_{i3}}^{+} (\nu_{13}, \bar{k}) \int \frac{d\bar{s}}{s^2} \tilde{\Phi}_{k_{i2}}^{+} (\nu_{12}, \bar{Q} - \bar{k} - \bar{s}).$$  \hspace{1cm} (31)

Dewangan and Bransden (1982) have shown that an integral of a product of some function $f(\bar{p})$ and the Fourier transform for the Coulomb distortion factor $\Phi_{iji}^{+}(\nu_{ij}, \bar{p})$ can be written as:

$$\int dp f(\bar{p}) \tilde{\Phi}_{iji}^{+} (\nu_{ij}, \bar{p}) = -4\pi Z_i Z_p f_c^{(+)(\nu_{ij})}$$

$$\times \lim_{\epsilon \to 0} \int dp f(\bar{p}) \frac{f(\bar{p})}{p^2 (p^2/2m_{ij} + \bar{p} v_{ij} - i\epsilon)} \left(1 + \frac{2\bar{p} \bar{k}_{ij}}{p^2}\right)^{\nu_{ij}}$$

if $f(\bar{p}) \to 0$ when $\bar{p} \to 0$. Using this property together with the Fourier convolution theorem we get for the amplitude $f_{pe}^{c2}$:

$$f_{pe}^{c2} = -\frac{\mu Z_p^2}{\pi^2} f_c^{(+)(\nu_{12})} f_c^{(+)(\nu_{13})} \lim_{\epsilon \to 0} \int dp \frac{|T_{\bar{f}_i}(\bar{Q} - \bar{p}) - T_{\bar{f}_i}(\bar{Q})|}{p^2 |\bar{Q} - \bar{p}|^2 (p^2/2m_{13} + \bar{p} v_{13} - i\epsilon)}$$

$$\times \left(1 + \frac{2(\bar{Q} - \bar{p}) |\bar{k}_{12}|}{|\bar{Q} - \bar{p}|^2}\right)^{\nu_{12}} \left(1 + \frac{2(\bar{Q} - \bar{p}) |\bar{k}_{13}|}{|\bar{Q} - \bar{p}|^2}\right)^{\nu_{13}}.$$  \hspace{1cm} (33)

The part of the scattering amplitude that describes the projectile-recoil ion (PT) interaction is derived directly from equation (14):

$$f_{pt} = -\frac{2\mu Z_p Z_i}{(2\pi)^6} \int dp \tilde{\Phi}_{k_{i2}}^{+} (\nu_{12}, \bar{p}) \int \frac{d\bar{s}}{s^2} \tilde{\Phi}_{k_{i3}}^{+} (\nu_{13}, \bar{Q} - \bar{p} - \bar{s}) T_{\bar{f}_i}(\bar{p}).$$  \hspace{1cm} (34)

We can simplify the integration over $\bar{s}$ in the $f_{pt}$ using the property of the Fourier transform that the Coulomb distortion factor $\tilde{\Phi}_{k_{i3}}^{+}$ has a sharp peak around zero when its argument approaches zero. Hence, we can factor out the slowly varying form factor $T_{\bar{f}_i}$ at the point $s = \bar{Q} - \bar{p}$. Moreover, if we use the Fourier convolution theorem along with the property (33), we obtain:
Furthermore, in the case of moderate and fast ion-atom collisions, the biggest contribution in integration over \( \vec{p} \) come from the region of small momentum \( \vec{p} \) then the slow varying Coulomb distortion factors may be factored out of the integral at \( \vec{p} = 0 \) and we can neglect \( p^2 \) terms in the denominator. Taking the z-axis along the velocity of scattering particle \( \vec{v}_{13} \), the denominator in \( f_{pt}^2 \) and \( f_{pt} \) may be expressed as

\[
\frac{p^2}{2m_{12}} + \vec{p} \cdot \vec{v}_{12} - i\varepsilon = \frac{1}{2m_{12}}((\vec{p} + m_{12} \vec{v}_{23})^2 - m_{12}^2 \vec{v}_{23}^2) + \vec{p} \cdot \vec{v}_{13} - i\varepsilon \approx p_z \vec{v}_{13} - i\varepsilon \quad (36)
\]

finally leading to the expression for scattering amplitude for practical calculations:

\[
f_{dir} = f_{pc}^1 + f_{pc}^2 + f_{pt} = K_{dir} f_{dir}^1(\vec{Q}) - \frac{\mu Z_p^2}{\pi^2 v_{13}} f_c^{(+)}(v_{12}) f_c^{(+)}(v_{13}) \left( 1 + \frac{2\vec{Q} \cdot \vec{k}_{12}}{Q^2} \right) \quad (37)
\]

\[
\times \lim_{\varepsilon \to 0} \int dp_z \left[ T_{f1}(Q - p) - T_{f1}(\vec{Q}) \right] - \frac{\mu Z_p^2}{\pi^2 v_{13}} f_c^{(+)}(v_{12}) f_c^{(+)}(v_{13}) \left( 1 + \frac{2\vec{Q} \cdot \vec{k}_{13}}{Q^2} \right) \quad (37)
\]

\[
\times \lim_{\varepsilon \to 0} \int dp_z \frac{Z_t T_{f1}(p)}{p^2 |Q - p|^2 (p_z - i\varepsilon)}.
\]

**2.3.3 SECOND BORN APPROXIMATION**

The second Born approximation has been often used for electron impact ionization calculations, however it is less commonly used for fast ion impact ionization of atomic systems. This may be explained by the fact that for years the main interest in studying heavy ion impact ionization was centered on the total or double differential cross sections as a function of ejected electron parameters. For these particular cross sections, the main contribution comes from collisions with small energy transfer or soft collisions, but the differential cross sections as a function of scattering angle can
hardly be described using soft collision models for larger scattering angles [2,3].

The first and second-order terms in Born perturbation expression are given by:

\[
\begin{align*}
    f_{\text{dir}} &= f_{\text{dir}}^{b1} + f_{\text{dir}}^{b2} \\
    &= -\frac{\mu}{2\pi} \langle \psi_{k23}^{(-)} \exp(i\vec{K}_f \vec{R}) | V_i | \varphi \exp(i\vec{K}_i \vec{R}) \rangle \\
    &\quad - \frac{\mu}{2\pi} \langle \psi_{k23}^{(-)} \exp(i\vec{K}_f \vec{R}) | V_i \tilde{G}_0^{(+)}(E) V_i | \varphi \exp(i\vec{K}_i \vec{R}) \rangle
\end{align*}
\]

and \( \tilde{G}_0^{(+)}(E) \) is a Green's operator with the spectral representation

\[
\tilde{G}_0^{(+)}(E) = \frac{1}{(2\pi)^3} \lim_{\varepsilon \to +0} \int d\vec{K} \sum_{\alpha} \frac{|\varphi_\alpha \exp(i\vec{K} \cdot \vec{R}) \rangle \langle \varphi_\alpha \exp(i\vec{K} \cdot \vec{R})|}{E - E_\alpha - \frac{K^2}{2\mu} + i\varepsilon}.
\]

The second Born term \( f_{\text{dir}}^{b2} \) represents a two-step interaction where the excitation of the target, by projectile, to some intermediate state \( \alpha \) is followed by ionization by the projectile from that intermediate state to the continuum. Therefore, we have to sum over all possible intermediate states \( \alpha \) including the continuum states as well in the Green's operator (39). Further analysis of the \( f_{\text{dir}}^{b2} \) shows that due to orthogonality of the wavefunctions of excited \( \alpha \) states and the wavefunction of the ground state \( \varphi_i \), all other contributions from intermediate states other than ground state from the \( V_{\text{mu}} \) part of interaction potential vanish. This interaction, however, determines the behavior of the DDCS at large scattering angles [2,3] and the leading-order contribution provides the ground state, i.e. \( \alpha = i \). Thus, the following expression for the second Born amplitude can be used for practical calculations:

\[
\begin{align*}
    f_{\text{dir}}^{b2} &\approx -\frac{\mu}{(2\pi)^4} \lim_{\varepsilon \to +0} \int d\vec{K} \langle \psi_{k23}^{(-)} \exp(i\vec{K}_f \vec{R}) | V_i | \varphi_i \exp(i\vec{K}_i \vec{R}) \rangle \\
    &\quad \times \frac{\langle \varphi_i \exp(i\vec{K} \cdot \vec{R}) | V_i | \varphi_i \exp(i\vec{K}_i \vec{R}) \rangle}{E - \frac{K^2}{2\mu} + i\varepsilon},
\end{align*}
\]

and it describes the two-step ionization mechanism, the elastic scattering of the projectile by an atomic system followed by the ionization of the atom via the projectile-electron interaction.

Considering that the three-body wavefunction (7) includes the final-state interaction beyond the first Born approximation, it can easily be shown that in the limit of
high collisional velocities the two amplitudes (38) and (41) derived from three-body model and second Born approximation may be compared.

Namely, in the limit of high collisional velocities \( u_i \gg u_e \) the peaking amplitude (28) asymptotically approaches the amplitude in first Born approximation

\[
f_{pe}^{c1} \rightarrow f_{dir}^{b1}(\vec{Q}), \tag{41}
\]

and the sum of the amplitudes \( f_{pe}^{c2} \) and \( f_{pt} \) from the equation (38) can be expressed as

\[
f_{pe}^{c2} + f_{pt} \rightarrow -\frac{\mu^2 Z_p^2}{(\pi^2)^2 K_i} \lim_{\epsilon \rightarrow 0} \int d\vec{p} \frac{[T_{fi}(\vec{Q} - \vec{p}) - T_{fi}(\vec{Q}) + Z_i T_{fi}(\vec{p})]}{p^2|\vec{Q} - \vec{p}|^2(p_z - i\epsilon)}. \tag{42}
\]

On the other hand, in the closure approximation the second Born term in equation (39) with an average energy for the target states \( E_{av} = 0 \), one can easily show that

\[
f_{dir}^{b2}(E_{av} = 0) = -\frac{\mu^2 Z_p^2}{\pi^2 K_i} \lim_{\epsilon \rightarrow 0} \int d\vec{p} \frac{[Z_i T_{fi}(\vec{Q}) + T_{fi}(\vec{Q} - \vec{p}) - T_{fi}(\vec{Q})]}{p^2|\vec{Q} - \vec{p}|^2(p_z - i\epsilon)}. \tag{43}
\]

Considering that the main contribution to the integral over \( \vec{p} \) comes from small momenta, the non-peaking terms in the three-body model reduce to the second Born amplitude in the closure approximation with an average energy for the target states of \( E_{av} = 0 \), i.e. \( f_{pe}^{c2} + f_{pt} \rightarrow f_{dir}^{b2}(E_{av} = 0) \) [4].

For calculation of the higher order terms we use the Sokhotsky theorem [20,21]

\[
\lim_{\epsilon \rightarrow \pm 0} \int \frac{f(x)}{x - x_0 \pm \epsilon} dx = P \int \frac{f(x)}{x - x_0} dx \pm i\pi f(x_0), \tag{44}
\]

where \( P \) stands for Cauchy principal value integral. Second order term \( f^{b2} \) can then be written as

\[
f^{b2} = f^{b2\, off} + f^{b2\, on}. \tag{45}
\]

The term that corresponds to Cauchy principal value integral is in atomic and nuclear scattering theory referred to as an off-shell term, while the second term or the pole in 44 is usually called as an on-shell term. The contributions from off-shell terms are usually regarded as energy non-conserving, because the principal value contribution specifically excludes the on-shell contribution at \( E = E_0 \) by allowing
short-lived quantum fluctuations in the intermediate energy. The calculation of offshell terms is a computationally costly task. Therefore, they are usually omitted from calculations of scattering amplitudes, under the pretense that they are not important. There have been several demonstrations of the significance of off-shell terms [22,23], thus they are included in our calculations (see Appendix A).

2.4 RESULTS AND DISCUSSION

We have carried out the calculations of DDCS for single ionization of hydrogen by 75 keV proton impact for fixed energy losses of $\Delta E = 30$, 40, 50 and 53 eV. The cross sections have been calculated using both the three-body method and the second Born approximation, with the numerical integration for higher-order amplitudes $f_{pe}^2$, $f_{pt}$ and $f_{de}^2$. Before going any further with the presentation and analysis of our calculations we give a brief overview of the experimental measurements.

2.4.1 EXPERIMENT

Our calculations are compared with the experimental results of Schulz et al. [1]. The kinematically complete experiment on single ionization, as said previously, requires evaluation of the momentum vectors of the all three collisional fragments, the projectile, the ejected electron, and the recoiled ion. It means that we need to measure two out of three momenta in coincidence, and the third momentum is then determined by using the law of momentum conservation. A method of measuring the scattered projectile and recoil-ion momenta directly has recently been performed [24–26] for light-ion impact at intermediate energies. The schematic of the experiment is shown in Fig. 2.

This type of experiments are also known as COLTRIMS or COLd TaRget Recoil Ion Momentum Spectroscopy. The proton beam, produced with hot cathode ion source and accelerated to 75 keV is crossed with atomic hydrogen beam generated by a microwave dissociator and cooled to about 5 K. The recoil ions are extracted, from the collision region, perpendicular to the incident projectile beam by a weak, nearly uniform electric field. After that they drift free and are detected by a two-dimensional position-sensitive detector. The scattered projectiles pass through a switching magnet, which cleans up the beam from components neutralized by capture from the target gas or the residual gas in the beam line. They are then decelerated and energy analyzed by an electrostatic parallel-plate analyzer and detected by a
FIG. 2: Experimental setup on single ionization measuring the momentum vectors of the scattered projectile and recoil ion.

2.4.2 RESULTS

In Fig. 3 to Fig. 6 the DDCS are plotted for fixed energy losses of $\Delta E = 30, 40, 50$ and $53 \text{eV}$ as a function of the scattered angle $\theta_p$. To understand the three-body dynamics we need to comprehend the role of different interactions in ionization process. It should be noted that any interaction included in the final state wavefunction is conceptually treated to all orders of perturbation theory. In practice it is not possible to find exact wavefunctions so higher order contributions may not be treated completely and/or accurately. On the other hand, any interaction that is only included in the operator is treated to whatever order Born series is expanded.
In principle, each order can be treated accurately, as well as the interaction included in the operator. Now the question is what is more important to include the various interactions to as many order as possible or treat specific higher-order contributions as accurate as possible. The two interactions that present a major challenge to the theory are the projectile-residual target interaction and the post collisional interaction between the outgoing projectile and ejected electron. Therefore, we are going to focus our discussion on the role of these two interactions.

FIG. 3: Double differential cross section for fixed energy loss of $\Delta E = 30$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semiclassical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).
FIG. 4: Double differential cross section for fixed energy loss of $\Delta E = 40$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).
FIG. 5: Double differential cross section for fixed energy loss of $\Delta E = 50$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).
FIG. 6: Double differential cross section for fixed energy loss of $\Delta E = 53$ eV as a function of projectile scattering angle. The experimental data are shown in solid blue circles [1]. The calculations are denoted as follows: dotted curve, CDW-EIS-Semi-classical [1]; dashed curve, SBA equation (43); dash-dotted curve, 3C, equation (28); solid curve, SBA-C, equation (38).
From Fig. 3 to Fig. 6 one can see that double differential cross sections fall off rapidly with the increasing scattering angle, which is an usual angular dependence for most processes. Also, the angular dependence is not very sensitive to energy loss up to 50 eV. At 53 eV the width of the DDCS suddenly decreases considerably. The significance of this value is that it corresponds to an ejected electron speed equal to the projectile speed, and it is well established that the effects due to PCI maximize at these speeds. What is also interesting about the calculations and experimental data presented here, is that different models, even though they all contain conceptually the same physics, differ very much from each other. This implies that the extent to which the various higher-order contributions are described in the different models is important in reproducing the experimental DDCS. In the three-body model we developed for the proton impact ionization of the atomic hydrogen both PT and PCI are included in the final state wave function for three-body Coulomb system (7), like in three-coulomb wave (3C) model [11,12,27] denoted as the dashed curve. Furthermore, we have shown that the three-body amplitude asymptotically approaches the second Born amplitude in the limit of high collisional velocities, thus as in second Born approximation (SBA), denoted with dash-dotted curve. The PT interaction is accounted for in the transition operator as well. Hence, we will refer to this model as second Born approximation - Coulomb waves (SBA-C) and it is denoted as the red solid curve in Fig. 3 to Fig. 6. The Coulomb distorted wave - eikonal initial state model (CDW-EIS), denoted as the dotted curve, is conceptually similar to the 3C model. Higher-order contributions from the projectile-electron interaction are treated in terms of a distortion of the ejected electron wave by the projectile in the final state and in terms of an eikonal phase factor in the initial state. However, the PT interaction is accounted for in terms of the eikonal approximation assuming a classical straight-line trajectory of the projectile [28], so it is also referred to as semi-classical model or CDW-EIS-SC.

The results presented in Fig. 3 to Fig. 6 indicate that the SBA-C model reproduces the shape of the $\theta_p$ dependence of the measured DDCS. At the energy losses of $\Delta E = 30$ eV and 40 eV, though, it seems to be a discrepancy of about 50 % in the magnitude between our calculations and experimental data. This may not be necessarily significant for our model because some uncertainties were introduced in the normalization of the experimental data [1]. Overall, the SBA-C model yields to the best agreement with experimental data for $\Delta E = 30$ eV to 50 eV. At $\Delta E = 53$ eV it
still agrees better with experimental data compared to the other calculations, but it does not describe the magnitude and width of the angular distribution of the DDCS. To understand the importance of PT and PCI interaction, we will first consider $\Delta E = 30 \text{ eV}$, see Fig. 3, that corresponds to the largest $|v_e - v_p|$, which means that the influence of PCI should be minimized in this case.

From Fig. 3 we see that both 3C model and CDW-EIS-SC, which accounts for the PT interaction in the initial or final state wavefunction underestimate the DDCS at intermediate and large angles. The problem is that at large $\theta_p$ all three particles approach each other to relatively small distances for PT interaction to produce significant deflection of the projectile, and the ejection of electron requires a close encounter with the projectile as well. But the 3C wavefunction, as it has already been stated, is only accurate when all the three particles are well separated. This implies that treating PT interaction in the asymptotic three-Coulomb wavefunction, or classically through eikonal approximation, may result in some inaccuracies. On the other hand, if the perturbation of the collision is not to large, then the magnitudes of higher expansion terms decreases with increasing order. So it may be feasible if we account for PT interaction in the transition operator, as in SBA and SBA-C methods. Indeed we see that the SBA results, which do not account for PCI, asymptotically approach both the experimental data and SBA-C calculations with the decreasing $\Delta E$, i.e with minimizing PCI effects.

Regarding the PCI, it is known that it distorts the asymptotic final state wavefunction. Since the projectile and the electron attract each other and their relative speed is small, they interact for a long time in the exit channel. Therefore, it is expected that higher-order terms are significant for the accurate description of collisional dynamics and it may be more appropriate to describe the PCI effects in terms of a final-state Coulomb wave. To closely analyze these issues, we consider the case where $\Delta E = 53 \text{ eV}$, shown in Fig. 6. In this case one might expect that the PCI effects are maximized, due to very small $|v_e - v_p|$, and that the PT interaction plays only a minor role. We see though from Fig. 6 that even the calculations that include the higher-order terms of the PCI, like 3C and DCW-EIS-SC have problem reproducing the DDCS without the accurate treatment of PT. This means that it cannot be ruled out that the focusing effect in the exit channel is based on an interplay between PCI and PT interaction.

Overall, the presented SBA-C model has the best agreement with experimental
data. The problem that the model has in reproducing the narrowing of the angular distribution of the DDCS at $\Delta E = 53$ eV may be attributed to both accuracy of the three-body final state wave function when $|v_e \approx v_p|$ and three particles in the exit channel are not well separated and the large numerical sensitivity of the calculations when $|v_e - v_p|$ is very small. The success of the SBA-C model suggests that the PT interaction is best accounted for in the operator of a second-order term of the transition amplitude. It seems that the terms beyond the second order are not significant, at least for this collisional system. We have also learned from the SBA-C model that the higher order contributions of PCI are important for the description of this interaction, and that is more appropriate to treat it in the final state wave function. An ultimate test of the theoretical description of the many-body dynamics in atomic collisions would be the measurements of fully differential cross sections for proton impact ionization of atomic hydrogen, which have only recently been initiated [1].
CHAPTER 3

MANY-BODY PROBLEM: SIMULTANEOUS SCATTERING

The discovery of the energetic particle emission from radioactive materials has initiated a new era in many-body physics. As soon as scientists realized that these particles can penetrate the matter, the idea was born to use charged particles to unravel the secrets of the matter, its constituents and governing forces. Extensive research in this area led to vast number of applications from material modifications and material analysis, to radiation therapy, fusion research, and accelerator science. The charged particle interactions with the surrounding matter are dictated by the collisions with the target atoms that the particle is undergoing while traversing through the material. Due to the combined collisional effects the charged particle experiences both angular and energy straggling. There has been significant activity during the last century in both theory and experiment to describe these collective effects. The main mechanism of energy loss of charged particle interacting with material is the energy transfer from projectile to target electrons due to inelastic processes, while the angular straggling is governed by the elastic scattering of projectile off of heavy target nuclei. The first mechanism has been extensively studied in past centuries and it resulted at the beginning of 1930s in the famous Bethe-Bloch formula for energy loss per distance traveled of relativistic projectiles passing through matter. The non-relativistic treatment of the same problem was offered in the 1960s by Lindhard, Schraff and Schiott [29], the so-called LSS-theory later revisited by Ziegler et al. [30]. The angular straggling was investigated during the 1950s, in the work of Molière [31], Scott [32], Goudsmit and Saunderson [33] and Lewis [34]. The four theories are mathematically closely related and are based on analytical treatment of the multiple Coulomb scattering of charged particles in matter. The multiple scattering or better yet the simultaneous scattering phenomenon is the one of main interests in our many-body study here.
3.1 SIMULTANEOUS VS. MULTIPLE SCATTERING

Multiple scattering historically refers to successive events that charged particles undergo while traveling through matter that change their direction of motion. These events are considered to be statistically independent and successive collisions of the charged particles with the target particles are treated as two-body or binary collisions. The influence of the other target particles is either accounted for through different approximations or completely neglected. The process of multiple scattering can be divided into three regions: low energy electron scattering, which has application in solid state and plasma physics; large angle scattering, at moderate energies and the high-energy; and small-angle multiple scattering, which is of practical importance in accelerator science. The latter has been extensively studied [31–34], and the developed theories are still in use with some refinement in simulation codes like GEANT4. These theories calculate the angular and spatial distributions of the charged particles after some length traveled through the material. Over time, the research in this area advanced in two directions. For practical applications either target is considered and the calculation of stopping powers of different materials or how the charged particle beam is altered after passing through the material. In addition, in accelerator science the collective effects in the bunch itself are making the problem of multiple scattering even more interesting. Historically looking, these calculations were state-of-the-art fifty years ago. Today, with more powerful computers available, the question is why not study all the collective effects at the same time. We have thus defined the term of simultaneous scattering, which refers to the charged particle scattering off of the total potential of all the target particles and all the other particles in the bunch at all times. Thus, the main difference between the two concepts is weather and how the surrounding target centers and the rest of the charge in the bunch influence the projectile motion through the target material. To be able to understand which effects are more important and under which conditions, we started with a single particle scattering off of all the target centers that are either frozen or moving. Finally, we considered the scattering of multiple bunches, accounting for the interaction among the particles in the bunch as well.
3.2 CLASSICAL MODEL AND SIMULATION

For our model we considered the classical case of scattering between a moving ion and a stationary target atom. In this classical picture, while the moving charge passes, the stationary particle recoils and absorbs energy. The energy transfer depends on the mass and charge of the colliding particles and initial velocity of the projectile. However, we have utilized quantum mechanics to calculate the interaction potential of target atoms or molecules. The simplest case of collisions, as we previously stated, are the ones involving atomic hydrogen. This allows us to study the simultaneous scattering effects free of any complications introduced by the complexity of the target atoms. The interaction potential for hydrogen atom is straightforward, and can be done analytically. For more complex atoms and molecules, one may use Hartree-Fock and Molecular Orbital methods.

3.2.1 INTERACTION POTENTIAL

We start with the positively charged ion $Z_p$ scattering off hydrogen atom as shown in Fig. 7.

FIG. 7: Scattering schematics of a positively charged ion colliding with atomic hydrogen.
The Hamiltonian of this system expressed in atomic units

\[ H = -\frac{1}{2\mu} \nabla_{r_1}^2 - \frac{1}{2\mu} \nabla_{r_2}^2 + \frac{Z_p}{r_1} - \frac{1}{r_2} - \frac{Z_p}{|r_1 - r_2|} \]  

(46)

where \( \mu \) is the mass of the projectile in atomic units and \( \vec{r}_1 \) and \( \vec{r}_2 \) are position vectors defined as in Fig. 7. The total energy of the system is then

\[ E_{\text{tot}} = E_1 + E_k = -\frac{1}{2} + \frac{1}{2\mu} k^2 \]  

(47)

a sum of energy of ground state of hydrogen atom \( E_1 \), assuming that the target atom is in ground state and kinetic energy of the incoming projectile \( E_k \). In the classical scattering case in the asymptotic region \( r_1 \gg r_2 \) the wave function must satisfy the asymptotic boundary condition, which represents the incident projectile moving with respect to ground state target atom

\[ \psi(\vec{r}_1, \vec{r}_2) \sim F_1(\vec{r}_1) \varphi_{100}(\vec{r}_2), \ r_1 \to \infty. \]  

(48)

The function \( F_1(\vec{r}_1) \) combines the incident plane wave and outgoing spherical wave

\[ F_1(\vec{r}_1) \sim \exp(i \vec{k}_1 \cdot \vec{r}_1) + f_1 \frac{\exp(i \vec{k}_1 \cdot \vec{r}_1)}{r_1} \]  

(49)

with scattering amplitude \( f_1 \) that carries all the information about collision.

The Schrödinger equation for this collisional system is then

\[ [H - E_{\text{tot}}] \psi(\vec{r}_1, \vec{r}_2) = 0, \]  

(50)

or expanded we may express it as

\[ \left[ -\frac{1}{2\mu} \nabla_{r_1}^2 - \frac{1}{2\mu} \nabla_{r_2}^2 + \frac{Z_p}{r_1} - \frac{1}{r_2} - \frac{Z_p}{|r_1 - r_2|} - E_1 + E_k \right] \psi(\vec{r}_1, \vec{r}_2) = 0. \]  

(51)

Multiplying equation (51) by \( \varphi_{nlm}^*(\vec{r}_2) \) and integrating it with respect to \( \vec{r}_2 \) we obtain

\[ \int d\vec{r}_2 \varphi_{nlm}^*(\vec{r}_2) \left[ -\frac{1}{2\mu} \nabla_{r_1}^2 - \frac{1}{2\mu} \nabla_{r_2}^2 + \frac{Z_p}{r_1} - \frac{1}{r_2} - \frac{Z_p}{|r_1 - r_2|} - E_1 + E_k \right] \times \]  

(52)

\[ \times F_1(\vec{r}_1) \varphi_{100}(\vec{r}_2) = 0. \]
Since we assume that the target atom is initially in the ground state \( \varphi_{nim} \rightarrow \varphi_{100}(\vec{r}_2) = \sqrt{\frac{1}{\pi}} \exp(-r_2) \) and knowing the Schrödinger equation for electron bound in hydrogen atom \([- \frac{1}{2} \nabla^2 - \frac{1}{r_2} - E_1] \varphi_{100}(\vec{r}_2) = 0 \) we are obtaining

\[
\left[ - \frac{1}{2\mu} \nabla^2 - \frac{1}{2} k_1^2 \right] F_1(\vec{r}_1) + Z_p(\varphi_{100}(\vec{r}_2)) \frac{1}{r_1} - \frac{1}{|\vec{r}_1 - \vec{r}_2|} \varphi_{100}(\vec{r}_2) F_1(\vec{r}_1) = 0. \tag{53}
\]

The second term in equation (53) represents the interaction potential of hydrogen in ground state

\[
V_{11} = Z_p(\varphi_{100}(\vec{r}_2)) \frac{1}{r_1} - \frac{1}{|\vec{r}_1 - \vec{r}_2|} \varphi_{100}(\vec{r}_2) = \tag{54}
\]

\[
Z_p \int \varphi_{100}^*(\vec{r}_2) \frac{1}{r_1} \varphi_{100}(\vec{r}_2) d\vec{r}_2 - Z_p \int \varphi_{100}^*(\vec{r}_2) \frac{1}{|\vec{r}_1 - \vec{r}_2|} \varphi_{100}(\vec{r}_2) d\vec{r}_2.
\]

Working out the above formula we are getting

\[
V_{11} = \frac{Z_p}{r_1} \int \int d\Omega \int \varphi_{100}^*(\vec{r}_2) \varphi_{100}(\vec{r}_2) r_2^2 d\vec{r}_2 \tag{55}
\]

\[
- Z_p \int \int d\Omega \int \varphi_{100}^*(\vec{r}_2) \frac{1}{|\vec{r}_1 - \vec{r}_2|} \varphi_{100}(\vec{r}_2) r_2^2 d\vec{r}_2.
\]

Using the expansion of the function

\[
\frac{1}{|\vec{r}_1 - \vec{r}_2|} = \sum_{i=0}^{\infty} \frac{(r <)^i}{(r >)^{i+1}} P_i(\cos \theta), \tag{56}
\]

and the assumption that the atom is in the ground state \( l = 0 \), the interaction potential becomes

\[
V_{11} = \frac{Z_p}{r_1} - 4Z_p \int \int r_2^2 \exp(-2r_2) d\vec{r}_2 + \int r_2 \exp(-2r_2) d\vec{r}_2 \tag{57}
\]

Using the integration by parts method we have finally the interaction potential of hydrogen atom

\[
V_{11} = -Z_p \left( \frac{1}{r_1} + 1 \right) \exp(-2r_1), \tag{58}
\]

that we will use in our calculations.
3.2.2 SIMULTANEOUS SCATTERING ALGORITHM - SESAME

The Simultaneous Scattering Algorithm (SESAME) is a simulation program designed to track the charged particles undergoing simultaneous scattering on multiple target centers. Since we are interested in close-range dynamics of the particles we based it on a particle-particle (PP) method for N-body simulations.

The particle-particle method is straightforward, it accumulates forces from the surrounding particles acting on the projectile and integrates the equations of motion in each time step. Direct integration approach in the Particle-Particle method is fairly but it comes with a high computational cost. It scales like \( N^2 \), or in other words, \( N^2 \) operations are necessary to evaluate the forces on \( N \) particles.

The SESAME supports simulation of the target sample with specific density and ionization degree. The target centers are considered as point charges and their positions are generated completely at random in orthogonal cell to simulate gas or plasma target. The size of the cell is defined by the number of particles in the target and target density.

The particle dynamics simulations starts with the calculation of the force \( F_i \) acting on the \( i^{th} \) projectile from surrounding target atoms

\[
F_i = m_i \frac{d^2 \vec{r}_i}{dt^2} = \sum_j^N \nabla V_{ij} =
\]

\[
= \begin{cases} 
\sum_j^N Z_p \left( 2 + \frac{\vec{r}_{ij}}{r_{ij}} \right) \exp(-2r_{ij}), & \text{for target neutrals} \\
\sum_j^N Z_p \frac{1}{r_{ij}}, & \text{for target ions}
\end{cases}
\]

The second order equation of motion of each projectile can be rewritten into a system of coupled differential equation of first order

\[
\frac{d\vec{r}_i}{dt} = \vec{v}_i
\]

\[
\frac{d\vec{v}_i}{dt} = \frac{F_i}{m_i}.
\]

The system of equations is solved using a Runge-Kutta of fourth order method. The solution returns the change in the projectile position, velocity and acceleration over a finite time step. The smaller the time step, the more accurate is the solution of the equations of motion but higher computational cost. To obtain the optimal time
step for both accuracy and computational time we used the Rutherford scattering mode of simulation and comparison between numerical and analytic solution.

After the projectile has been moved, the time counter is increased by time step and the simulation continues by recalculating the forces at the new projectile position, see Fig. 8. Once the particle exits the target, we record its position and velocity and scattering angle, which we then use to determine the angular and spatial distributions of our projectile beam.

FIG. 8: SESAMe scheme of operation; In blue - target neutrals, in red - target ions, in black - projectiles.

The SESAMe is a single particle tracking code but it also supports beam tracking as well. The beam particles are propagated at the same time and mutual repulsion due to the Coulomb interaction among the projectiles in the bunch was accounted for in the total force acting on the projectiles.
Simulation also allowed for the recoil events in the target. In the same manner, as for a projectile, the force acting on a target atom from a projectile was calculated using the derived potential (57) and the fourth order Runge-Kutta method is used for solving the equations of motion for all the target atoms in each time step. The energy straggling of the projectiles due to energy transfer to target atoms was recorded giving us the energy distribution of our projectile beam once it travels through the target.

The test to the simulation was the Rutherford scattering case for which the analytic solution exists and the law of conservation of energy. For more details see Appendix B.

3.3 SIMULTANEOUS SCATTERING EFFECTS

The interaction of the charged particle beam with matter depends on the properties of both the target and the projectile. Thus, the simultaneous scattering effect will be sensitive to varying target and projectile parameters. We considered the change in density and degree of ionization of the target, as well as mass and initial energy of the projectile beam. We have also studied the effects of target atoms recoil and the interaction between particles in the bunch. Our aim is to determine the conditions for which the simultaneous effects are important and therefore can not be omitted from the models and theories. We used nonlinear curve fitting, to be able to compare the angular distributions obtained from numerical calculations for both multiple binary and simultaneous scattering. The best fit was the Voigt profile, which is the convolution of Gaussian and Lorentzian. This profile is common for processes which are dominated by collisions, and it is referred to as pressure broadening or collisional broadening. To extract more information about the differences between the two sets of data, we also used q-q plots. The q-q plot is a statistical non-parametric technique for the comparison of two distributions, or data sets. The q in the name stands for quantile, which is basically the value of the variable for which the certain percentage of data is below that value. So, if the two distributions are the same, then they will have the same values for the quantiles, and when they are plotted against each other they should fall on the straight line \( y = x \). Any departures in following this line indicates a difference between the two data sets (see Appendix A).
3.3.1 TARGET DENSITY

By changing the target density from $\rho = 10^{22}$ cm\(^{-3}\) (liquids) to $\rho = 10^{26}$ cm\(^{-3}\) (inertial confinement plasma) we have observed, as shown in Fig. 10 - 15, that the simultaneous scattering effect becomes more significant with increasing density. By increasing the target density, the inter-atomic distance is decreased. The inter-atomic distance now is comparable or smaller than the scattering length, so the influence of surrounding atoms becomes stronger. The q-q plots have more prominent S shape with higher density, telling us that the data quantiles plotted on the ordinate (here simultaneous scattering angles) are heavy-tailed and narrow-peaked compared to the data which quantiles are plotted on the abscissa (here multiple binary scattering). It appears that the surrounding target particles are compensating each other's force on the projectile, focusing the beam at small angles, like in Fig.9, but due to the increase in the value of the total force, due to close range interactions with target particles, there are more particles scattered at larger angles than in the case of multiple binary scattering.

FIG. 9: The change of the direction of motion of a particle due to the change of its momentum's transverse component.
\[ 75 \text{ keV } p^+, \rho \sim 10^{22} \text{ cm}^{-3} \]

**FIG. 10:** The simultaneous scattering effect for the target density of \( \rho = 10^{22} \text{ cm}^{-3} \).

**FIG. 11:** The q-q plot for simultaneous scattering effect for the target density of \( \rho = 10^{22} \text{ cm}^{-3} \).
75 keV $p^+$, $\rho \sim 10^{24}$ cm$^{-3}$

**FIG. 12:** The simultaneous scattering effect for the target density of $\rho = 10^{24}$ cm$^{-3}$.

**FIG. 13:** The q-q plot for simultaneous scattering effect for the target density of $\rho = 10^{24}$ cm$^{-3}$.
FIG. 14: The simultaneous scattering effect for the target density of $\rho = 10^{26}$ cm$^{-3}$.

FIG. 15: The q-q plot for simultaneous scattering effect for the target density of $\rho = 10^{26}$ cm$^{-3}$. 
3.3.2 TARGET IONIZATION

With the ions present in the target, the multiple interactions of the projectile with target atoms are affected by long range Coulomb interaction as well as the screened interaction. This means that the projectile will interact with ions in the target on the longer scales. The long range of the Coulomb force suggests that simultaneous effect when the ions are in the target is going to be more prominent. This can be observed in Fig. 16 - 19, and the more evident stretched S shape in the q-q plot agrees with these observations.

FIG. 16: The simultaneous scattering effect for the 10% ionized target.
FIG. 17: The q-q plot for simultaneous scattering effect for the 10% ionized target.

75 keV $p^+$, $\rho \sim 10^{26}$ cm$^{-3}$, 100 % ions

FIG. 18: The simultaneous scattering effect for the 100% ionized target.
3.3.3 PROJECTILE MASS

The dependence on projectile mass is also observed in simultaneous scattering effects. The heavier more inert projectiles tend to stay on their initial paths when passing through targets, while projectiles with smaller mass, the more mobile ones tend to experience more angular straggling. We considered a low energy muon beam passing through dense hydrogen target $\rho = 10^{26}$ cm$^{-3}$ and what is noticed is more pronounced simultaneous scattering effect. This tells us that the mobile muons are more affected by the focusing from the surrounding particles and that even small compensations in force correct their path through target. Also they tend to scatter on larger angles than heavier protons which agrees with more pronounced departure from the straight line at large angles in q-q plot, Fig. 21.
FIG. 20: The simultaneous scattering effect for the 8 MeV $\mu^+$.

FIG. 21: The q-q plot for simultaneous scattering effect for the 8 MeV $\mu^+$.
3.3.4 PROJECTILE VELOCITY

At the higher projectile energies we observed, Fig. 23 - 26, as expected, that the simultaneous scattering effect is smaller. Here the difference in transverse momentum due to simultaneous scattering is compensated with the high momentum in the direction of motion in the multiple binary scattering case, as shown in Fig. 22. Now the simultaneous effect is less noticeable. With higher projectile energies the two angular distributions become more similar, thus the q-q plots follow the $y = x$ line more closely.

![Diagram showing the change in direction of particle motion due to the change of its momentum's longitudinal component.](image)

FIG. 22: The change in direction of particle motion due to the change of its momentum’s longitudinal component.
7.5 MeV $p^+$, $\rho \sim 10^{26}$ cm$^{-3}$

- Simultaneous
- Multiple Binary

$\Delta$FWHM < 5%

FIG. 23: The simultaneous scattering effect for the 7.5 MeV $p^+$.

FIG. 24: The q-q plot for simultaneous scattering effect for the 7.5 MeV $p^+$. 
FIG. 25: The simultaneous scattering effect for the 750 MeV $p^+$.

FIG. 26: The q-q plot for simultaneous scattering effect for the 750 MeV $p^+$.
3.3.5 TARGET RECOIL

We have also considered how the simultaneous scattering effect is influenced by the energy transfer from the projectile to target atoms due to recoil events. What is observed in Fig. 27 is that, the momentum transfer to target atoms is smaller in the case of simultaneous scattering. This supports our idea that the opposite forces from target atoms acting on projectile cancel out, more focusing the beam. This implies that the projectile motion is less affected by the energy transfer in the simultaneous scattering case, resulting in more focused angular distribution of the beam after traveling through target.

FIG. 27: The simultaneous scattering effects dependence on the momentum transfer due to the recoil of target atoms.
FIG. 28: The simultaneous scattering effects dependence on the momentum transfer due to the recoil of target atoms.

3.3.6 PROJECTILE BUNCHES

Practical applications call for projectile bunches. If only one charged particle is enough for applications, there would be no need for accelerator science. Therefore, it is more realistic in simulations to track particle bunches rather than single particles. The SESAMe supports bunch tracking and it includes the mutual Coulomb repulsion between the particles in the beam. The mutual repulsion, interestingly enough, results in narrowing of the angular distributions of the 75 keV proton beam, as we may see from Fig. 29. This seems counter intuitive at first, since one would expect that the Coulomb repulsion would contribute to the beam heating. This question is particularly interesting for the process of muon cooling, where the interaction of the muon beam with matter is used to reduce the transverse momentum of the beam through inelastic processes in the target, while at the same time reconstructing the longitudinal momentum with applied accelerating field, Fig. 22. This results in increased luminosity of the beam, which is needed for practical applications.
FIG. 29: The simultaneous scattering effects compared for single particle and bunch tracking, 75 keV $p^+$. The data obtained for more mobile 15 MeV positive muon bunch illustrates this phenomenon the best. The evolution of this generated muon bunch was followed in time and space under different conditions. We have tracked the muon bunch in vacuum, with particles interacting within the bunch trough the Coulomb potential, and in a liquid hydrogen target ($10^{22} \text{ cm}^{-3}$), which is used for the practical muon cooling. We have also looked into the bunch evolution in the target when the interaction between the particles is not included as in the single particle tracking case. From Fig. 30 one can observe that collective effects from particles interacting in the bunch and the heating of the beam due to projectiles interaction with the target atoms compensate each other, producing less broad beam's angular distributions.
This indicates that the particles in the bunch are highly correlated and that the particles exchange energy, scattering one off another. When they are given some initial velocity the energy is exchanged more in the direction of motion resulting in larger spatial spread in longitudinal direction. It is only in plasma that we observe this level of correlations and collective effects, so we may actually say that the charged particle bunch is a one component plasma object.

This comprehensive study of simultaneous scattering showed us that when considering one particle scattering on multiple targets, the simultaneous scattering effect, under certain conditions, may be up to 20%. Now, the question is what are the conditions in the problem we have at hand, and is this 20% significant. For energetic heavy ion particle beam passing through sparse targets this effect may not be relevant, for instance 300 MeV/c muon beam passing through liquid hydrogen. On the other hand, for some kinematic models of inertial confinement plasma, it may be quite important. When considering the simultaneous scattering of the particles inside the bunch off of one another and on the target particles, it seems that the effect is much more pronounced and it needs careful attention.
FIG. 30: Evolution of the 15 MeV $\mu^+$ bunch. Different colored dots correspond to spatial distributions of the same bunch at different times.
CHAPTER 4

MANY-BODY PROBLEM IN PLASMA: TOMOGRAPHY

In the field of plasma physics, a main consideration is the complex interaction of many charged particles with self-generated or external electromagnetic fields. This unique entanglement is what makes plasma physics a fascinating field for basic research. Because of self-consistent motion of particles inside the plasma, the plasma object is full of instabilities, and nonlinear phenomena. Given its nature, the plasma state is characterized by many body dynamics and a complexity that vastly exceeds the ones exhibited in the solid, liquid, or gaseous states. Correspondingly, the study of plasma properties is one of the most far ranging and difficult research areas in physics today.

The experimental techniques available for plasma characterization allow us to measure only integrated effects of collective plasma behavior. These type of measurements are necessary for evaluating macroscopic plasma parameters like rotational and electron temperatures and electron densities, which are important for practical applications. However, they do not tell us much about the internal dynamics of the observed plasma object. Like in collision studies, we need differential data to accompany integral ones, for the understanding of the underlying collective processes, and for testing the available theoretical models.

Plasma emission tomography is a way of transforming the integral data into the spatial population distributions of the plasma constituents. It may serve scientists as a magnifying glass to look at the internal dynamics of the plasma object without disturbing it.

Tomography reconstruction is a well developed field. Medical application was the driving force behind this research area for many decades now. The field of applied mathematics during this time has produced state of the art reconstruction methods and algorithms, while computer science and engineering followed with their practical implementations. In medical tomography the development of new reconstruction methods was followed by new measurement technologies, leading to most advanced medical diagnostics.
The idea to use tomography for physics problems, such as diagnostics of plasma or charged particle beams, has been around for some time now as well. However, in this case scientists face a challenge of implementing the tomography methods for specific problems due to limited amount of data, and signal noise. This introduced a gap between the state-of-art reconstruction methods and real physical application. This is the reason why most of the publications in plasma tomography are demonstrations of implementations of very simple reconstruction methods. It is analogous to using the simple Simpson rule for the integration of all possible functions.

Thus, different tomography methods should be used for the reconstruction of local plasma parameters, depending on the properties of the studied plasma object and the amount of information needed. Our objective here is to apply refined numerical methods for plasma tomography in order to study a plasmoid-like structure observed in an ongoing experiment at the Department of Physics, at Old Dominion University. The revised numerical methods are tested on analytic functions for which the reconstruction function is analytically solvable.

We used the emission tomography, since the plasma is a strongly radiating object, thus there is no need for perturbing the system for the purpose of taking measurements. We measured the integrated intensity of intrinsic plasma emission in emission tomography.

The emissivity of a plasma object is described by its volume emission coefficients, $\varepsilon$, which depend on various plasma parameters. Tomography allows us to more fully use the information contained in the emission of an object. In particular, we can determine the spatial distributions of excited states, ion and electron density, and temperature.

The equation for the radiative transfer along the line $L$ in plasma is given by

$$\frac{dI_\nu}{dl} = \varepsilon_\nu(l) - \kappa_\nu(l)I_\nu(l)$$  \hspace{1cm} (61)

where $\varepsilon_\nu$ and $\kappa_\nu$ are respectively, the local emission and absorption factors at a fixed radiation frequency $\nu$; $l$ - the coordinate along the line $L$.

Solving this equation for the emergent radiation $I_\nu(p, \theta)$ can be written as

$$I_\nu(p, \theta) = \int_{-\infty}^{\infty} dl \varepsilon_\nu(l) \exp \left[ - \int_{l}^{\infty} \kappa_\nu(l')dl' \right].$$  \hspace{1cm} (62)

Integral plasma emission $I_\nu(p, \theta)$ (projection) is recorded along a system of direct rays, see Fig. 33 in a direction defined by $\theta$ and distance $p$ from the origin.
In the approximation of optically thin plasma \((\kappa \equiv 0)\), which is true in our case, registered radiation is described by the classical integral Radon transform of spatial distribution function of emission coefficients [35]. Correspondingly, the spatial distribution of emission coefficients may be determined using the tomography methods from recorded spectral line intensities of observed plasma object.

4.1 PLASMOID

In the microwave supersonic flow discharge of pure argon, the plasmoid-like structure was observed as a secondary downstream phenomenon coupled to the microwave cavity [36]. The plasmoid appears to be sustained by a low-power surface wave which propagates along its surface and the surface of the containing quartz tube. The propagation of the waves in plasma is a many-body effect, which makes the problem of plasmoid formation very interesting for our study here. Our initial analysis led to the conclusion that the plasmoid formation may be caused by aerodynamic effects in the supersonic flow [36]. However, we further observed that the plasmoid’s position relative to the cavity is constant, which indicates that the plasmoid is also an effect of collective plasma behavior. We used several computer tomography methods for characterization of this plasma object to study its collective behavior.

4.2 EXPERIMENT

The experimental set-up where the plasmoid formation occurs, shown in Fig. 31, is a combination of supersonic flow tube and a microwave cavity discharge. In an evacuated quartz tube at pressures of 1 — 3 Torr, supersonic flow was generated with a Mach 2 cylindrical convergent-divergent (de Laval) nozzle, upstream from the cavity. To sustain a cylindrical cavity discharge at power density between 0.5 and 4 W/cm\(^3\), we used a commercial microwave generator, operating in the S-band at 2.45 GHz. Argon gas was fed into the stagnation chamber through a gas manifold. The gas flow was established by using a roots blower in conjunction with two roughing pumps. The capacity of the pumping system allowed a supersonic flow to be generated in a pressure range of 1 — 20 Torr.
We used optical emission spectroscopy as our primary diagnostic tool to observe the spectra of the excited states of argon. Optical emission spectroscopy is a simple, non-intrusive, in situ diagnostic technique, where the subjects of analysis are the wavelength and intensity of the radiation emitted during the transitions from the higher energy (excited) level to the lower energy level of atoms and molecules. An Automated Measurement System (AMS) was built with the aim to increase the overall precision of the taken measurements as well as to streamline the measurement process. It consists of a mirror and a microcontroller-based system, composed of two high-precision stepper motors and several sensors providing precise feedback control. The AMS controlled the angle and distance of the measuring system from the cylindrical cavity, within sub-degree angle precision, and sub-millimeter distance precision. A CCD camera was used in conjunction with a spectrometer for spectral line detection. The measurements were taken at position 3.5 cm from the cavity,
which corresponds to the mid-section of the plasmoid. The experimental set-up allowed us to record projections under 21 angles in the range from 48 to 168 degrees, and 17 projections for each angle, with a sampling rate of 0.2 cm across the diameter of the quartz tube.

Two spectral lines were used for determining the population of argon excited states at 706 nm and 714 nm. The two lines correspond to the Ar I \[3s^23p^5(^2P_{1/2})4p \rightarrow 3s^23p^5(^2P_{3/2})4s\] transition, for \(J_i - J_k\) \((2 - 2)\) and \((2 - 1)\) respectively, and they were calibrated to black body radiation using the Spectra-Physics Quartz Tungsten Halogen Lamp. First, the measured intensities of the given spectral
lines were multiplied with the irradiance per count calculated from the black body calibration. This gave us irradiance in units of \([\text{mW}/\text{m}^2(\text{nm})]\). Then, we introduced the length of the plasma region (about 2 cm) and expressed the irradiance \((P_x)\) in terms of radiometric quantities \([\text{W}/\text{cm}^3\text{(nm)}]\). Irradiance is converted to photonic quantities using the following relation

\[
\frac{dN}{dt} = P_x \lambda 5.03 \times 10^{15} \left[ \frac{ph}{\text{cm}^3\text{s}} \right].
\]  

(63)

The population is then,

\[
N_u = \frac{dN}{A_{ul}} \left[ \frac{1}{\text{cm}^3} \right]
\]  

(64)

where \(A_{ul}\) is the transition probability and \(g_u\) is the statistical weight of the upper excited state. We used these population in our calculation to retrieve the information about the spatial distribution of neutrals in plasmoid to understand the plasma collective behavior and determine if it is actually governed by a surface wave.

4.3 TOMOGRAPHY

The origin of the word tomography comes from the Greek words \(\tau\ ομος\) (tomos) - a slice, a piece, or a cut, and \(\gamma\ ρω\) (grapho) - to draw, or to write. Essentially tomography is just that, drawing a slice. A tomography imaging system produces a cross sectional image of an observed object. Two dimensional tomography reconstructs the object, with spatial distribution \(f(\mathbf{r}) = (x, y) \in \mathbb{R}^2\) from the measured values of its angular projections \(g(p, \theta)\), see Fig. 33. It is widely used as a diagnostic tool in medical imaging (Computer Assisted Tomography, or CAT scan). Since the 1960's there has been increased activity in the area of plasma physics to use tomography for plasma diagnostics [37–40] and since the 1980's tomography became a valuable diagnostic tool for characterization of accelerator beams [41–44]. Beam physics as well as plasma, due to their nature, provide an invaluable insight in the collective effects. The so-called reconstruction problem, where the internal structure or some property of an internal structure is determined without interfering and/or damaging the object, is of particular interest for many-body physics.
FIG. 33: The tomographic projection $g(p, \theta)$ is the collection of parallel ray integrals through the object $f$ in the direction specified by $\theta$; the Radon transform is the set of all such angular projections for $\theta \in [0, \pi)$

4.3.1 RADON TRANSFORM

In two dimensions, the mapping of a given function $f(x, y)$ defined by the projections or the line integrals of $f$ along the all possible directions defined by $\theta_i$, is described by the Radon transform $R\{f\}$ [45], provided that the integral exists. Namely,

$$\tilde{f} = R\{f(x, y)\} = \int_L f(x, y) \, dl,$$

where $L$ is the line of integration and $dl$ is the increment of the length along that line.
This transform as well as its inverse was first studied by Johann Radon (1917), who showed that if \( f \) is continuous and has compact support than both the radon transform \( \mathcal{R}\{f\} \) and the inverse radon transform \( \mathcal{R}^{-1}\{f\} \) of the function are uniquely determined. To better visualize the reconstruction problem it is useful to transfer to new coordinates \( s \) and \( p \) that are rotated by angle \( \theta \) in respect to the \( x \) and \( y \) coordinates, as shown in Fig. 33. Then

\[
\begin{align*}
  p &= x \cos \theta + y \sin \theta, \\
  s &= -x \sin \theta + y \cos \theta,
\end{align*}
\]

and the line integral (65) now depends on the values of the \( p \) and \( \theta \)

\[
\tilde{f}(p, \theta) = \mathcal{R}\{f\} = \int_L f(x, y)dl.
\]

If \( \tilde{f}(p, \theta) \) is known for all \( p \) and \( \theta \), then \( \tilde{f}(p, \theta) \) is the two-dimensional Radon transform of \( f(x, y) \).

The coordinates \( x \) and \( y \) in the new rotated coordinate system are given as:

\[
\begin{align*}
  x &= s \cos \theta - p \sin \theta, \\
  y &= s \sin \theta + p \cos \theta,
\end{align*}
\]

giving more explicit form for the transform

\[
\tilde{f}(p, \theta) = \mathcal{R}\{f\} = \int_{-\infty}^{\infty} f(s \cos \theta - p \sin \theta, s \sin \theta + p \cos \theta) \, ds.
\]

We are going to present the explicit calculation of the Radon transform on the example of a Gaussian distribution function, Fig. 34

\[
f(x, y) = \exp(-x^2 - y^2).
\]
Using the coordinate transformation we can easily show that $x^2 + y^2 = s^2 + p^2$, then we have for a Radon transform

$$
\tilde{f}(p, \theta) = \int_{-\infty}^{\infty} \exp(-p^2 - s^2) \, ds \\
= \exp(-p^2) \int_{-\infty}^{\infty} \exp(-s^2) \, ds \\
= \sqrt{\pi} \exp(-p^2).
$$

Hence we have as the final result:

$$
\Re\{f(x, y)\} = \Re\exp(-x^2 - y^2) = \sqrt{\pi} \exp(-p^2).
$$

With a good sampling technique of the input data and *Newton-Cotes quadrature* numerical integration we obtained the Gaussian distribution function projections.
The problem of reconstruction, as we previously stated, is a major topic of interest in this work. During the past few decades, it has become very important, due to its vast application from medical imaging to radio-astronomy, plasma diagnostics and geophysical exploration.

Mathematically the problem may be defined as follows. In order to recover the desired information about the internal structure of the observed object, we need to invert the Radon transform to solve for function $f$ in terms of its projections $\hat{f}$:

$$f(x, y) = \mathcal{R}^{-1}(\hat{f}(p, \theta) + \eta),$$

(73)
where \( \eta \) is the noise introduced by the measurements. In the new coordinate system (67) we can express our unknown function \( f(s,p) \) through its two-dimensional Fourier transform

\[
f(s,p) = \int_{-\infty}^{\infty} F(\nu_s, \nu_p) \exp[i2\pi(s\nu_s + p\nu_p)] \, d\nu_s d\nu_p .
\] (74)

The expression for the projection \( \hat{f}(p, \theta) \) can be now rewritten as

\[
\hat{f}(p, \theta) = \int_{-\infty}^{\infty} f(s,p) \, ds = \int_{-\infty}^{\infty} F(\nu_s, \nu_p) \, d\nu_s d\nu_p \int_{-\infty}^{\infty} \exp[i2\pi(s\nu_s + p\nu_p)] \, ds \quad \text{(75)}
\]

\[
= \int_{-\infty}^{\infty} \exp[i2\pi p\nu_p] F(\nu_s, \nu_p) \, d\nu_s d\nu_p \int_{-\infty}^{\infty} \exp[i2\pi s\nu_s] \, ds \, \delta(\nu_s)
\]

and using the properties of delta function \( \int_{-\infty}^{\infty} f(x) \delta(x) dx = f(0) \) we finally get the projection formula

\[
\hat{f}(p, \theta) = \int_{-\infty}^{\infty} \exp[i2\pi p\nu_p] F(0, \nu_p) \, d\nu_p .
\] (76)

Here we use the Fourier slice theorem, which relates the one dimensional Fourier transform of a projection at an angle \( \theta \), \( \hat{F}_\theta(\nu_p) \) of our unknown function to the central slice, at angle \( \theta \), of its two-dimensional Fourier transform \( F(0, \nu_p) \).

The projection \( \hat{f}(p, \theta) \) may now be expressed as

\[
\hat{f}(p, \theta) = \int_{-\infty}^{\infty} \exp[i2\pi p\nu_p] \hat{F}_\theta(\nu_p) \, d\nu_p .
\] (77)

If we now look at the unknown function in polar coordinates in frequency domain

\[
f(x,y) = \int_{0}^{2\pi} d\theta \int_{0}^{\infty} d\nu \nu F(\nu, \theta) \exp[i2\pi \nu (r \sin(\varphi - \theta))] = \int_{0}^{2\pi} d\theta \int_{0}^{\infty} d\nu \nu F(\nu, \theta) \exp[i2\pi \nu p_0] + \int_{0}^{\pi} d\theta \int_{0}^{\infty} d\nu \nu F(\nu, \theta + \pi) \exp[i2\pi \nu p_0] ,
\] (78)

and using the property \( F(\nu, \theta + \pi) = F(-\nu, \theta) \) [45] the above expression for \( f(x, y) \) may be written as
\[ f(x, y) = \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} d\nu \nu |F(\nu, \theta) \exp[i2\pi \nu p_0| = \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} d\nu \nu |\hat{F}(\nu) \exp[i2\pi \nu p_0| = \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} d\nu \nu |\hat{F}(\nu) \exp[i2\pi \nu p_0| \]

where argument \( p_0 \) corresponds to the point \((x, y)\) and \( r \) and \( \phi \) are polar coordinates in XY plane.

The inverse Fourier transform applied on the above equation (80) then returns the unknown function

\[ f(x, y) = \int_0^{\pi} d\theta \int_{-\infty}^{\infty} F^{-1}(\nu) \hat{f}(p, \theta) dp . \]  

Knowing the inverse Fourier transform of the function \(|\nu|\) [46]

\[ F^{-1}(\nu) = -\frac{1}{2\pi^2 \nu^2} \]  

we finally obtain the inverse Radon formula

\[ f(x, y) = -\frac{1}{2\pi^2} \int_0^{\pi} d\theta \int_{-\infty}^{\infty} \frac{\hat{f}(p, \theta)}{(p - p_0)^2} dp . \]  

Integration by parts of the equation (82) gives another well known formula for inverse radon transform

\[ f(x, y) = -\frac{1}{2\pi^2} \int_0^{\pi} d\theta \int_{-\infty}^{\infty} \frac{\partial \hat{f}(p, \theta)}{\partial p} \frac{1}{(p - p_0)} dp , \]  

where \( \frac{\partial \hat{f}(p, \theta)}{\partial p} \) is a partial derivative of the object's projections.

**4.4 INVERSION METHODS**

Having access to these inversion formulas, (82) and (83), one would think that our job is done, but in the reality it is only a beginning for an applied problem.

The basic inversion formulas are rigorously valid if \( f \) is continuous with compact support, and the projections \( \hat{f} \) are given for all the angles, meaning that infinite set of projections is needed rather than a discrete set, which is the case for practical applications. There is a theorem by Smith, Solomon and Wagner [47], which states that a function \( f \) of compact support in \( R^2 \) is uniquely determined by any infinite set, but not by any finite set, of its projections. Thus, it seems that we need to
sacrifice uniqueness for practical applications. In other words, we can only find a good enough, non-unique approximation. Possible alternatives approaches are to either impose appropriate a priori conditions on a solution, or to utilize a large number of projections [48]. Another point to consider is the stability of the solution in respect to the noise introduced by the measurement technique. Thus, we may be in need of extremely precise measurements, that may be physically impossible to perform, to obtain a satisfactory estimate of our unknown function. These issues have been extensively studied in past three decades, resulting in number of different algorithms and various numerical approaches. Here, we present several different numerical approaches that were used for the two-dimensional plasma tomography.

4.4.1 DIRECT INTEGRATION

In the direct integration method we start from the basic inversion formula (82) and using the high accuracy numerical interpolation and integration we reconstruct the unknown function $f$. However, the integral (82) has a singularity at $p = p_0$ and it needs a special treatment. From the mathematical analysis we know that the integrals (82) and (83) correspond to the Cauchy principal value integrals [49]

$$
\int_{-\infty}^{\infty} \frac{\varphi(x)}{x^{2m}} = \int_{0}^{\infty} dx x^{-2m} \left\{ \varphi(x) + \varphi(-x) - 2 \sum_{k=1}^{m} \frac{x^{2k-2}}{(2k-2)!} \varphi^{(2k-2)}(0) \right\},
$$

$$
\int_{-\infty}^{\infty} \frac{\varphi(x)}{x^{2m+1}} = \int_{0}^{\infty} dx x^{-2m-1} \left\{ \varphi(x) - \varphi(-x) - 2 \sum_{k=1}^{m} \frac{x^{2k-1}}{(2k-1)!} \varphi^{(2k-1)}(0) \right\}.
$$

In our particular case $m = 1$ and by simple substitution $x = p - p_0$ from the above equation we obtain two different formulas for the inversion Radon formula

$$
-\frac{1}{2\pi^2} \int_{0}^{\pi} d\theta \int_{-\infty}^{\infty} \frac{\tilde{f}_\theta(p)}{(p - p_0)^2} = 
$$

$$
= -\frac{1}{2\pi^2} \int_{0}^{\pi} d\theta \int_{0}^{\infty} dx \frac{\tilde{f}(x + p_0) + \tilde{f}(-x + p_0) - 2\tilde{f}(p_0)}{x^2},
$$

$$
-\frac{1}{2\pi^2} \int_{0}^{\pi} d\theta \int_{-\infty}^{\infty} \frac{\partial_p \tilde{f}(p)}{(p - p_0)} = 
$$

$$
= -\frac{1}{2\pi^2} \int_{0}^{\pi} d\theta \int_{0}^{\infty} dx \frac{\partial_p \tilde{f}(x + p_0) - \partial_p \tilde{f}(-x + p_0)}{x},
$$

which are the starting point of our calculations, see Appendix D.
To test our method we are going to use the two-dimensional Gaussian distribution function \( f(x, y) = \exp(-x^2 - y^2) \), shown in Fig. 34 that has an analytic solution of the equation (82). We start by generating the discrete set of its projections \( \hat{f}_\theta(p) = \sqrt{\pi} \exp(-p^2) \), as shown in Fig. 35, and use a direct integration method to obtain the original distribution Fig. 36.

This method is straightforward, but it needs to be used with caution due to its high sensitivity to the noise introduced by measurements. It seems that some information may be lost in the reconstruction process due to lack of filtering. Thus, smoothing and interpolating of data beforehand may be required when using this method.
FIG. 36: The Inverse Radon transform of Gaussian distribution function
\[ \Re \{ \exp(-x^2 - y^2) \} = \sqrt{\pi} \exp(-p^2) \] using the direct integration method.
4.4.2 FILTERED-BACK PROJECTION

The filtered-back projection (FBP) algorithm is widely used because it has been shown to be more accurate and more suitable for fast implementation. Here we present the FBP for parallel projection data with the \( \tau \) sampling interval.

We start with the equation for the inverse Radon transform (80) that may also be written as

\[
f(x, y) = \int_0^\pi Q_\theta(-x \sin \theta + y \cos \theta) d\theta,
\]

where \( Q_\theta \) expressed in the frequency domain is

\[
Q_\theta(p) = \int_{-\infty}^{\infty} \tilde{F}_\theta(\nu) |\nu| \exp(i2\pi \nu p) d\nu,
\]

and the \( \nu \) has dimension of spatial frequency.

Essentially what these formulas imply is that from each projection \( \tilde{f}_\theta(p) \) we need first to calculate a filtered projection \( Q_\theta(p) \) using (87) and then use (86) to reconstruct the unknown function \( f(x, y) \). In principle the integration in filtered projection (87) has to be carried over all the spatial frequencies. However, in practice the energy contained in the Fourier transform components above a certain frequency is negligible, so for practical purposes we may consider the projections bandlimited.

When the highest frequency in the projections is finite \( W = \frac{1}{2\tau} \) we may express equation (87) as

\[
Q_\theta(p) = \int_{-\infty}^{\infty} \tilde{F}_\theta(\nu) H(\nu) \exp(i2\pi \nu p) d\nu,
\]

where \( H(\nu) \) is a filter which purpose is to cut off higher frequencies since the higher frequency signals are attributed to the noise.

The most commonly used filters are the Ram-Lak filter [50]

\[
H(\nu) = \begin{cases} 
|\nu|, & |\nu| \leq W \\
0, & \text{otherwise},
\end{cases}
\]

and the Shepp-Logan filter [51], Fig. 37

\[
H(\nu) = \begin{cases} 
|\nu|^{\sin(\nu)}/\nu, & |\nu| \leq W \\
0, & \text{otherwise}.
\end{cases}
\]
$H(\nu)$ functions, actually, represent the Fourier transform of a filter with which the projection must be processed, and the impulse response $h(p)$ of this filter is then given with the inverse Fourier transform of $H(\nu)$ [50]

$$h(p) = \int_{-\infty}^{\infty} H(\nu) \exp(i2\pi \nu p) d\nu. \tag{91}$$

In the case of most practical applications, when we have a discrete set of projections, measured with the spatial sampling interval $\tau$, so $p = n\tau$, where $n$ is an integer, only the impulse response for the same sampling interval is needed

$$h(n\tau) = \begin{cases} \frac{1}{4\pi^2}, & n = 0 \\ 0, & n \text{ even} \\ -\frac{1}{n^2\pi^2\tau^2}, & n \text{ odd} \end{cases} \tag{92}$$

for Ram-Lak filter and

FIG. 37: Ram-Lak and Shepp-Logan filters in both frequency and spatial domain.
\[ h(n\tau) = \begin{cases} \frac{2}{\pi^2\tau}, & n = 0 \\ -\frac{2}{n^2\tau^2(4\tau^2 - 1)}, & n \neq 0 \end{cases} \quad (93) \]

for Shepp-Logan filter.

With this being said, now both projections \( f_\theta(p) \) and \( h(p) \) are band-limited and together with the convolution theorem

\[ Q_\theta(p) = \int_{-\infty}^{\infty} f_\theta(p')h(p-p') \, dp' \quad (94) \]

they produce the values of the filtered projections at the sampling points

\[ Q_\theta(n\tau) = \tau \sum_{k=-\infty}^{\infty} h(n\tau - k\tau)f_\theta(k\tau). \quad (95) \]

In practice, we can assume that each projection \( f_\theta(k\tau) \) is zero outside the index range \( k = 0, 1, ..., K - 1 \), thus we may express the filtered projection as

\[ Q_\theta(n\tau) = \tau \sum_{k=0}^{K-1} h(n\tau - k\tau)f_\theta(k\tau), \quad n = 0, 1, ..., K - 1. \quad (96) \]

Finally the reconstructed function \( f(x, y) \) may be obtained by the discrete approximation of the (86)

\[ f(x, y) = \frac{\pi}{K} \sum_{i=1}^{K} Q_{\theta_i}(-x \sin \theta_i + y \cos \theta_i), \quad (97) \]

where the \( K \) angles are the ones at which the projections are sampled. This means that each filtered projection has to be back-projected. For every point \((x, y)\), as we said, there is a point \( p = -x \sin \theta + y \cos(\theta_i) \) for a given angle \( \theta \). The contribution of each filtered projection \( Q_{\theta_i} \) to the reconstruction of \( f(x, y) \) at the particular point \((x, y)\) depends on the value of \( p \) for a given \( \theta_i \). Depending on a resolution of our reconstruction image it may happen that value of \( p = -x \sin \theta + y \cos(\theta_i) \) does not correspond to the values at which \( Q_{\theta}(p) \) was sampled. The suitable interpolation of \( Q_{\theta} \) values at such \( p \) successfully deals with that problem. The reconstruction of a Gaussian function using the FBP method is shown in Fig. 38, and they are in excellent agreement with the original Gaussian distribution function.
Filtered back-projection may be done in the frequency domain as well, using the Fast Fourier Transform algorithms. The advantage of doing so is a faster implementation of the discrete convolution (96). One has to be careful though, because in frequency domain only periodic convolutions can be performed, while in the case of (96) the convolution is aperiodic. This is resolved by zero padding (ZP). The frequency domain implementation of FBP is then expressed as

\[ Q_{\theta}(n\tau) = \tau \times \text{IFFT}\{\text{FFT}(f_{\theta}(n\tau) \text{with ZP}) \times \text{FFT}(h(n\tau) \text{with ZP})\} \] (98)

where FFT and IFFT represent Fourier transform and inverse Fourier transform respectively.

This type of algorithm is the one usually found in the inverse Radon transform routines in commercial software packages, such as MATLAB.

We have also tested FBP on the example of the two-dimensional Gaussian distribution function \( f(x, y) = \exp(-x^2 - y^2) \), shown in Fig. 34 which has analytic solution of the equation (82 ). Again, we have generated the discrete set of its projections \( f_\theta(p) = \sqrt{\pi} \exp(-p^2) \), as seen in Fig. 35, and used the FBP method to obtain the original distribution Fig. 38, more in Appendix E.
FIG. 38: The Inverse Radon transform of Gaussian distribution function $\mathfrak{R}\{\exp(-x^2 - y^2)\} = \sqrt{\pi} \exp(-p^2)$ using the Filtered Back Projection with Ram-Lak filter.
4.5 TOMOGRAPHIC ANALYSIS OF EXPERIMENTAL RESULTS

From the reconstructed two-dimensional Gaussian distribution, see Fig. 36 and Fig. 38, we observe that both inversion methods are quite sensitive to a number of angular projections and to the angular range on which the projections were sampled.

We used the smooth cubic spline approximation to smooth the noise in the measured signal of the plasmoid projections for the experimental data. The reconstructed populations of excited argon atoms are plotted in Fig. 39 and Fig. 40.

Fig. 39 and Fig. 40 show, that the direct integration method compared to FBP methods for the plasmoid gives different results. It seems that in direct integration some information about the observed object is lost. The fact that even smoothed data with the direct integration method is not returning the same population distributions of plasma emitters as the FBP method, indicates that the appropriate filtering technique of the high frequency noise signal is needed in plasma tomography. Thus we will consider the population distribution obtained from FBP as higher fidelity data in our analysis.

All of the above tells us that it is quite challenging to apply various advanced and well developed tomography algorithms in both applied math and engineering field to study a physical object like plasma, when one is limited in the number of measured angular projections and when noise signal is considerable.

The missing parts in the population distribution of FBP data, seen on Fig. 39 and Fig. 40, may be attributed to the small range and number of angular projections, limited by our experimental set-up. It is, however, obvious from the plots that the excited species are mainly concentrated at the rim of the plasmoid object. This observation indicates that the plasmoid is indeed sustained by a surface wave and that the collective effects play an important role in the plasmoid formation. However, we need more experimental data to entirely understand the discharge parameters in this region. Ongoing experiments on the plasmoid will help us to determine the temperature and concentration profiles throughout the plasmoid to fully characterize it and reveal the nature of the surface wave that sustains it.

Furthermore, we intend to explore alternative tomography algorithms that would minimize the noise induced tell-tale streaking and other artifacts that may appear in the reconstructed image when the number of angular projections of observed object is limited. This is of particular importance, since we hope to extend this research towards tomography of accelerator beams for both beam emittance measurements
and characterization of the bunches, to understand the underlying collective behavior. One direction in doing so would be to revisit the already proposed iterative algorithms, which are supposed to work well on limited set of data [41–44]. These methods designate a family of algorithms that adjust the values of the pixels in the reconstructed image until its projections most closely resemble the measured ones. The solution, however, is not unique, and it is necessary to establish a priori criteria for its convergence.
FIG. 39: Population of the Ar I $[3s^23p^5(^2P_{1/2})^4p \rightarrow 3s^23p^5(^2P_{3/2})^4s]$ at 706.72 nm, 2.4 Torr, reconstructed.
FIG. 40: Population of the Ar I \([3s^23p^5(2P_{1/2})^4p \rightarrow 3s^23p^5(2P_{3/2})^4s]\) at 714.70 nm, 2.4 Torr, reconstructed.
We also plan to go even further and develop a wavelet based tomography algorithm. The wavelet transform, unlike the Fourier transform, is localized in both frequency space and configuration space. The spatial delocalization effects in wavelets are greatly attenuated by the corresponding zeros of the wavelet functions (vanishing moments) [52–54]. Hence, wavelet based reconstruction algorithms tend to be localized spatially and can be applied to obtain reconstructions only when a limited number of angular projections are available [55]. Thus, it is essential to design a particular wavelet transform to suit each problem we have at hand, based on the understanding of the underlying physics processes.

Overall the development of new tomography algorithms and computational methods is essential for the understanding of many-body effects in both plasmas and charged particle beams.
CHAPTER 5

CONCLUSION

The area of many-body physics provides a framework for understanding the collective behavior of large number of interacting particles. Even though the underlying physical processes governing the motion of each individual particle may be quite simple, the study of the collective effects can be extremely complex, due to particle coupling and entanglement. The emergent phenomena, in some cases, may not even resemble the underlying fundamental laws of physics. The uniqueness of this research area is that the many-body aspects of every problem have to be approached in a specific way. We used three different approaches to study many-body problems that are relevant for both fundamental research and various applications.

Quantum mechanical tools were used to study the three-body dynamics in the process of proton impact ionization of atomic hydrogen. We have demonstrated that a fully quantum mechanical approach to a three-body problem developed from the first principles is needed for the fundamental understanding of three-body dynamics. Our results showed that for large angle scattering, the main contribution comes from the projectile target interaction, and that this effect is properly accounted for when treated in transition operator. The importance of post collisional interaction was also discussed. The PCI is known to distort the asymptotic final state wavefunction, due to the long range Coulomb interaction between the projectile and electron in exit channel. This means that the higher order terms of PCI are significant for explaining three-body dynamics, and that its effects are then best described in terms of a final state Coulomb wave. Further analysis indicated that the focusing effect in DDCS in the case when the ejected electron speed is equal to projectile speed, may be a result of an interplay between PCI and PT interaction. Recently initiated measurements of fully differential cross sections for proton impact ionization of atomic hydrogen should unravel this question and provide the ultimate test-bed for the theory.

Classical scattering theory was applied on the many body problem of a charged particle beam interacting with a material target. We considered the simultaneous scattering of many charged particles in a bunch on many target centers. The interaction of charged particles with neutral target atoms was described through a
screened potential, that was derived from first principles, while the charged particles interacted through Coulomb force with one another. We compared the effects of traditional multiple binary scattering with simultaneous scattering effect for different projectile and target properties. For that purpose we developed computational code, based on classical scattering, that included close range interactions between the particles. We observed the simultaneous scattering effects through the focusing of angular distributions of particle bunches. We attributed this effect to the mutual compensation of surrounding particle influences on the projectile. This problem of simultaneous scattering of particles interacting with multiple target centers is a very complex and highly dependent on the initial conditions of both projectile and target. Depending on the initial properties of projectile and target, we may have several competing processes that may mask the collective effects under certain conditions. The effect may not be significant for low to average target densities, while for very dense plasma systems it may be important, depending on the application. The most interesting result was obtained from the bunch tracking calculations where the simultaneous scattering was accompanied by mutual repulsion between particles in the bunch. The close range particle-particle treatment of the interaction between the particles in the bunch and target centers showed significant improvement in angular beam size, indicating strong correlations between the particles in the bunch. This correlation level resembles the collective behavior of plasmas, therefore we may conclude that the charged particle bunch may be considered as one component plasma. More detailed studies of the collective effects inside the bunches are needed for understanding of these interesting phenomena.

Finally, we used a semi-empirical approach to study the many-body effects in plasma. The main challenge in studying collective effects in plasma is the integral or average nature of measured data. The use of emission plasma tomography allows us to get more information about internal dynamics of the plasma object from the measured integral data. The difficulties in the implementation of very advanced tomography reconstruction methods to a real physical objects, are in the limited number of measurements available, and the noise in the measurement signal. What we observed is that the noise smoothing and proper filtering of data is highly necessary, because otherwise we can have some tell-tale streaking and phantoms in our reconstructed image that are non physical features in the observed object. We applied the refined
direct integration and filtered back projection method for tomographic reconstruction to study the collective effects responsible for formation of a plasmoid-like object in the afterglow of supersonic flow microwave discharge in pure argon. The population distributions obtained from tomography reconstructed emission spectroscopy data agree with the initial assumption that the plasmoid is sustained by a low-energy surface wave. Ongoing experiments should entail more details about the character and origin of this object.
BIBLIOGRAPHY


APPENDIX A

BEAR II

For calculations of ionization amplitudes for the process of proton impact ionization of atomic hydrogen, we have used the BEAR II computer code developed for single and multiple electron transition of helium. The code has been modified for the case of single impact ionization of hydrogen, with new wavefunctions and no correlation terms. In Fig. 41 the pseudo code for the BEAR II is presented.

1. Cowan module and Cwave mode calculates the wave functions that we are needed for scattering matrix elements.

2. Bmatrix module calculates matrix elements both on and off shell ones.

3. Sigma98 module calculates amplitudes and cross sections for the process in question, once supplied by proper wavefunctions and scattering matrix elements.
FIG. 41: Pseudo code for BEAR II.
APPENDIX B

SESAME

|=================================================================================================
| MODULES                                                                                           |
|=================================================================================================
| set the number of centers included in simulation                                                 |

module center

integer, parameter:: centers=1600
integer, parameter:: nt = 1600 ! # of particles in the target
integer, parameter :: test=0
| testing (test=1 and nt=1 -Rutherford Scattering, test=0 and nt=1 - Rutherford with recoil) |

end module center

|=================================================================================================
| program SESAME                                                                                   |
|=================================================================================================

use center

implicit none

|CONSTANTS                                                                                          |
|=================================================================================================
| double precision, parameter:: a0 = 0.53E-10 ! bohr radius                                         |
| double precision, parameter:: mm = 1.89E7 ! (a0) mm in atomic units                              |
| double precision, parameter:: pi = 3.141592653                                                    |
| double precision, parameter:: rad = 57.2958                                                     |
| double precision, parameter:: au = 0.511 ! MeV/c^2, mass of electron                            |
| double precision, parameter:: Eau = 27.211E-6 ! MeV energy atomic unit                         |
| double precision, parameter:: invalpha = 137.0 ! fine structure constant                        |
| double precision, parameter:: Na = 6.022E23 ! avogadro's number                                |

|PROJECTILE                                                                                         |
|=================================================================================================
| integer, parameter:: nb=100 ! 100 ! 10                                                            |
| integer, parameter:: np=10                                                                      |
| ! mass of the particles (amu=0.511 MeV.c^-2)                                                      |
| double precision, parameter:: mp = 206.7771                                                    |
| ! 1836.149                                        !                                              |
| ! 1                                                                                            |
| ! charge of the particle (amu=e-)                                                                 |
| double precision, parameter:: Zp = 1.0                                                          |
| ! initial projectile velocity (1 a.u.=alpha*c)                                                   |
| double precision, parameter:: V0 = 1.0                                                          |
| double precision xp(np),yp(np) ! projectile position, x-coordinate & y-coordinate                |
| double precision Vxp(np),Vyp(np)! projectile velocity, x-component & y-component               |
| double precision fxp(np),fyp(np)! forces on projectile, x-component & y-component               |

|TARGET                                                                                             |
|=================================================================================================
| double precision, parameter:: M = 1.007 ! (g/mol) molar mass of the target                        |
double precision, parameter:: at = 1836.2
double precision, parameter:: Zt = 1.0  ! charge of the particle
integer ion(nt)         ! ion or neutral info
double precision xt(nt,2)  ! projectile position/velocity, x-direction
double precision yt(nt,2)  ! projectile position/velocity, y-direction
double precision ftx(nt)  ! force in the x-direction on a target particle
double precision fty(nt)  ! force in the y-direction on a target particle
double precision dmin(nt),gmat(nt)  ! inter-particle distances
double precision,parameter :: dalpha = 180.0  ! molecular orientation

! MATERIAL PROPERTIES
double precision, parameter:: rhop = 4.08E19  ! part/nm3
double precision, parameter:: rho = 70.0E-3   ! (g/mm3) density of the target
integer, parameter:: aorm « 1 ! 1- atomic; 2-molecular target
double precision rho2D, power  ! density of the target in 2D
double precision r0, L, Ly     ! target dimensions
! minimal distances for the target particles
double precision rain, rion, rneutral
integer, parameter:: ions = 0  ! percentage of ions in target

! SOLVING
integer,parameter:: n=4
double precision t,ti, tf
double precision, parameter:: tmax=30.0 10.0
double precision, parameter:: hs = 0.0001 !E-20 10.00001 10.00000
integer, parameter:: sample=2000 1000 100 10000

external dfcn
real rand

! WORKING
integer times(8),hour, minute, sec
integer i,j,k,ki,kj,jk
double precision Tkp(np=nb),Vp(np=nb),Lzp(np=nb),Ep(np=nb)
double precision Tkt(nt),Vt(nt),Lzt(nt),Et(nt)
double precision bmin, bmax, db, b(nb), gamma(np=nb)
double precision theta(nb*np), thetaR, dsigma(nb),dsigmaR
double precision yinp(np*n), ylnt(nt*n), youtp(np*n)
double precision youtt(nt»n),yloc(np=nb),x,y

! FILES
OPEN(UNIT=7,FILE='mu+ g4 bunch SESAMe.dat')
OPEN(UNIT=6,FILE='projectile.dat')
OPEN(UNIT=9,FILE='target.dat')
OPEN(UNIT=10,FILE='lawsofconservations.dat')
OPEN(UNIT=11, FILE='mu+ g4 bunch scattering data.dat')
if(nt.le.2.and.nb.gt.3.or.test.eq.1) OPEN(UNIT=12,FILE='RutherfordTest.dat')
call marktime(0,times,hour, minute, sec)
do i=7,13,1
  if(i.eq.12.and.test.eq.1.or.i.ne.12) then
    write (1,108)
    write (1,109)times(3),times(2),times(1)
    write (1,110)times(5),times(6),times(7)
    write (1,108)
  end if
end do
write (11,105)
if(nt.eq.1.and.nb.gt.3.or.test.eq.1) write (12,107)

ti=0.0

! simulation parameters
rO = 74.14E-12/aO

if(nt.eq.1) then
rO=0.0
end if

! target
power = 2.0/3.0
rho2d= rho**power !(rho*Na/M)**power

rmin = 2.0*sqrt((mm-mm)/(pi*rho2d))
L = sqrt((mm**2)*float(nt)/rho2d)*1.5
Ly=L

write (7,300) test
write (7,108)
write (7,301) nb,np,mp,Zp,V0
write (7,108)
write (7,302) nt,centers,aorm,mt,Zt,dalpha
write (7,108)
write (7,303) M,rho,rho2D,ions,r0,rmin,L,Ly
write (7,108)
write (7,304) tmax,hs,n,sample

xp= 0.0
Vxp= V0
Vyp=0.0

call thetargets(13,aorm,nt,ions,r0,dalpha,rmin,L,ly,ion,xt,yt)

if(test.eq.1.or.nt.le.2) then
bmin=rO+0.0005
bmax=rO+1.5005
db= (bmax-bmin)/float(nb)

xp=-10.0
end if

do i=1,nb
write(*,'(bunch=','i
write(8,100)i
write(9,100)i

ti=0.0
ki=0.0

ki=np*(i-1)
if(test.eq.1.or.nt.le.2) then
b(i)=bmin+float(i-1)*db
end if

do j=1,np
kj=3*(j-1)+j

yinp(kj) = -0.25*Ly*rand() * xp(j)
yinp(kj+1) = 0.5*rand() - 0.25)*Ly*yp(j)
yinp(kj+2) = Vxp(j)
yinp(kj+3) = Vyp(j)

if (test.eq.1.or.nt.le.2) then
  yinp(kj+1) = b(i)
  yp(j) = b(i)
end if

gamma(j) = 1.0/((1.0-((Vxp(j)**2+Vyp(j)**2))/(invalpha)**2)**0.5)

end do

do j=1,nt
  kj=3*(j-1)+j

  yint(kj) = xt(j,1)
yint(kj+1) = yt(j,1)
yint(kj+2) = xt(j,2)
yint(kj+3) = yt(j,2)

  gamma(j) = 1.0/((1.0-((xt(j,2)**2+yt(j,2)**2))/(invalpha)**2)**0.5)
end do

write(10,102) i, sum(Tkp*amu*gamma)/(invalpha**2) + sum(Vp) + sum(Vt), Eau,
  sum(Lzp) + sum(Lzt)
write(10,103)
do while(ti.le.tmax)

  tf = ti + hs
  x = 0.0
  y = 0.0 ! center of projectile mass
  yloc = yinp

  call rk4n(dfcn, ti, tf, np, mp, Zp, nt, mt, Zt, yint, yinp, youtp, ion, n)
  call rk4n(dfcn, ti, tf, nt, at, Zt, np, mp, Zp, yint, yinp, youtp, ion, n)

  if(((k/sample)*sampla.eq.k)) then
    call L0fC(np, mp, Zp, nt, mt, Zt, youtp, Tkp, Vp, Lzp, ion, n)
    call L0fC(nt, mt, Zt, np, mp, Zp, yint, youtp, Tkt, Vt, Lzt, ion, n)
    call forces(np, mp, Zp, nt, mt, Zt, youtp, xfp, yfp, ion, n)
    call forces(nt, mt, Zt, np, mp, Zp, yint, youtp, ftx, fty, ion, n)
  end if

  gamma(j) = 1.0/((1.0-((youtp(kj+2)**2+youtp(kj+3)**2))/(invalpha)**2)**0.5)
  Tkp(j) = Tkp(j) * gamma(j)/(invalpha**2)
end do
Vp(j)=Vp(j)+Eau

write (8,101) ti,j,(youtp(jk),jk=kj,kj+n-1),fyp(j)=mp,fyp(j)=mp

end do

do j=1,centers

kj=(n-1)*(j-1)+j

gamm(j) = 1.0/((1.0-(((youtp(kj+2)**2+youtp(kj+3)**2)**0.5) & & /invalpha)**2)**0.5)

Tk(j)=Tk(j)*amu*gamm(j)/(invalpha**2)

Vt(j)=Vt(j)+Eau

write (9,101) ti,j,(yint(jk),jk=kj,kj+n-1),ftx(j)=mt,fty(j)=mt

end do

write (10,104)ti,sum(Tkp),sum(Vp),sum(Tkp+Vp),sum(Lzp), sum(Tkt),sum(Vt),sum(Tkt+Vt),sum(Lzt),sum(Tkp+Vp)+sum(Tkt+Vt)

end if

do j=1,mp

jk=(n-1)*(j-1)+j

x=x+youtp(jk)

y=y+youtp(jk+1)

end do

x=x/np ! x center of mass of projectiles

y=y/np ! y center of mass

if(nt.gt.2.and.(x.gt.L.or.(x.lt.0.0.and.ti.gt.10.0))) exit

!preparation for new step

yinp=youtp

yint=yint

ti=ti

k=k+1

end do

call L0fC(np,mp,Zp,nt,mt,Zt,youtp,yint,Tkp,Vp,Lzp,ion,n)
call L0fC(nt,mt,Zt,np.mp.Zp,yint,youtp,Tkt,Vt,Lzt,ion,n)
do j=1,mp

kj=(n-1)*(j-1)+j

theta(kj)=1.0/((1.0-(((youtp(kj+2)**2+youtp(kj+3)**2)**0.5) & & /invalpha)**2)**0.5)

theta(kj+1) = datan2(youtp(kj+3),youtp(kj+2))

write(11,106) j,j.theta(kj)=rad, (sum(Tkp*theta)+amu/(invalpha**2)) & & + sum(Vp)=Eau, youtp(kj), youtp(kj+1)-yp(j), youtp(kj+2), & & youtp(kj+3), (sum(Tkp*theta)*amu/(invalpha**2))

end do

end do

! for testing purpose

if(nt.le.2.and.nb.gt.3.or.test.eq.1) then
call sorting2(nb,theta,b)
call differentiation(nb,theta,b,dsigma,l)
do i=1,nb
thetaR=datan(Zp*Zt/(b(i)*mp*v0**2.0))


dsigmaR=((Zp*Zt/(2.0*mp*v0**2.0))**2.0)/((sin(thetaR))**4.0)
write (12,106) i,i,theta(i)*rad,b(i),dsigma(i),
& b(i)=abs(dsigma(i))/sin(theta(i)), &
& 100.0*abs((theta(i)-2.0*thetaR)/(2.0*thetaR))
end do

end if ! testing
call marktime(1, times, hour, minute, sec)
write (7,108)
write (7,109) times(3),times(2),times(1)
write (7,110) hour, minute, sec

stop

100 format (/,'# bunch =',14,'/6x,'t',10x,'par',10x,'X',10x,'y',16x,'Vx',16x,'Vy',&
& 16x,'fx',16x,'fy')
101 format (/43.0,s18.10,2e18.10)
102 format (/27.0,2e18.10,10x,2x,6e10)
103 format (7x,'t',16x,'Vp',16x,'Ep',15x,'Lzp',15x,'Vt',16x,'Et',&
& 'Es',16x,'Lzt','/')
104 format (e14.6,9e18.10)
105 format (1x,'bun',2x,'par',7x,'theta',14x,'E',17x,'x',17x,'y',17x,'Vx',15x,&
& 'Vy',15x,8x,'Kp')
106 format (214,7e18.10)
107 format (1x,'bun',2x,'par',7x,'theta',14x,'E',13x,'b',13x,db/dtheta,10x,dsigma, &
& 13x,'dsigma',10x,theta')
108 format ('***************************************************************/)
109 format ('**** ',12,' ',12,' ',14, ' ****')
110 format ('**** time: ',12,' ',12,' ',12,' ',12, ' **** */)
200 format ('t=',e14.6,3x,'i',3x,'ion',5x,'x',5x,'y')
201 format (214,2f12.6)
300 format ('**T',4x,'S',4x,'T',=,14)
301 format ('**T','R','O','J','E','C','T','I','L','E*','/','# bunch=',15.0, &
& '# part=',15.2x,'mp=',10.3x,'Zp=',f6.2x,'V0=',f6.2)
302 format ('**T','A','R','C','E','T*','/','# part=',15.2x,'# centers=',15.2x, &
& 'AorM=',15.2x,'st=',f10.3x,'T=',f6.2x,'alpha=',f6.2)
303 format ('**T','A','R','C','E','T*',/,'# part=',15.2x,'# centers=',15.2x, &
& '# part=',15.2x,'# centers=',15.2x, &
& 'Ax=',f10.3x,'x',2x,'Ly=',f6.3)
304 format ('**S','O','L','V','E*','R*','/','tmax=',f8.2x,'dt=',f12.6x,'neq=', &
& 14,2x,'samples',110)

end program

! SUBROUTINES

! subroutine marktime(key,t,hour, minute, sec) ! technical stuff
! print the date of calculations and elapsed time
! IN:
! key = 0 first call: print the current day and time
! key = 1  print elapsed time
! Comment:
! function date_and_time returns summer time (+1 hour in winter)
! Alex G.

implicit none

integer, parameter :: nout=6
integer key
integer ti(8), tf(8), t(8)
integer sdelta, hour, minute, sec
character(9), dimension (12), parameter:: months = &
("January","February","March","April","May","June"
,"July","August","September","October","November","December")

integer day, month, year

if (key==0) then
    call date_and_time(values=ti)
    write(nout,100) ti(3), months(ti(2)), ti(1)
    day=ti(3)
    month=ti(2)
    year=ti(1)
    hour=ti(5)
    minute=ti(6)
    sec=ti(7)
    !for the winter time minus one hour
    write(nout,101) ti(5), ti(6), ti(7)
    t=ti
else
    call date_and_time(values=tf)
    sdelta = (tf(5)-ti(5))*3600 + (tf(6)-ti(6))*60 + (tf(7)-ti(7))
    hour = sdelta/3600
    minute = (sdelta - 3600*hour)/60
    sec = sdelta - 3600*hour - 60*minute
    write(nout, 102) hour, minute, sec
    t=tf
end if

100 format('***** ',12,',',A,' ',14,' *****')
101 format('***** time: ',12,'.',i2,'',12,':',i2,' ',14,' *****')
102 format(,, ' elapsed time',/,' ',12,'.',i2,'',12,':',i2,'',12)

end subroutine marktime

end subroutine marktime

end

! subroutine positions(nout,ionornot,n,x,y,lmin, Dxx, Dyy)
! sets random positions of the particles in D^2 cell
! no two particle can be closer than lmin
! written by: Ana Samolov
!

! input ...
! nout - defines the output file unit
! ionornot - refers whether the particle is neutral(O) or ion(I)
! n - number of particles
! lmin - minimum distance allowed between particles
! Dxx & Dyy - dimension of the cell
!
! output ...
! x() & y() - arrays of positions generated for n- number of particles

implicit none

integer n, i, j, nout
integer ionornot, try, maxtries, tooclose
double precision lmin, Dxx, Dyy
double precision dx, dy, r
double precision x(n), y(n)
real rand

if (ionornot.eq.0) then
  OPEN (UNIT=nout+1, FILE="neutrals.dat")
  write(nout+1,202) lmin
  WRITE(nout+1,200)
endif

if (ionornot.eq.1) then
  OPEN (UNIT=nout+2, FILE="ions.dat")
  write(nout+2,202) lain
  WRITE(nout+2,203)
endif

maxtries = 1000

x(i) = rand()*Dxx
y(i) = rand()*Dyy

if (ionornot.eq.0) then
  WRITE(nout+1,201) x(i), y(i)
endif

if (ionornot.eq.1) then
  WRITE(nout+2,201) x(i), y(i)
endif

do i=2,n
  try = 0
  tooclose = 0
  do while (tooclose.eq.0)
    x(i) = rand()*Dxx
    y(i) = rand()*Dyy
    try = try +1
    do j=1,i-1
      dx = x(j) - x(i)
      dy = y(j) - y(i)
      r = (dx**2+dy**2)**0.5
      if (r <= lmin ) then
        tooclose = 0
        exit
      else if (j=(i-1)) then
        tooclose = 1
      end if
    end do
  end do
  if (try > maxtries) then
    write (*,*) 'The cell is too small'
  endif
endif
STOP
END IF

END DO

IF (IONORNOT.EQ.0) THEN
  WRITE(NOUT+1,201) X(I),Y(I)
END IF

IF (IONORNOT.EQ.1) THEN
  WRITE(NOUT+2,201) X(I), Y(I)
END IF

END DO

200 FORMAT (8X,'X0',11X,'Y0')
201 FORMAT (2F12.4)
202 FORMAT ('RMIN = ', F12.4)
203 FORMAT (8X,'X+',11X,'Y+')

END SUBROUTINE POSITIONS
y0=0.0

write(nout,400)
if (n.eq.1.and.aorm.eq.1) then !aorm=1 one atom
  tx(1,1)=0.0 !xt
  ty(1,1)=0.0 !yt
  tx(1,2)=0.0 !Vxt
  ty(1,2)=0.0 !Vyt
  ionornot(1)=ionss/100
end if
if (n.eq.2) then !one molecule (two particles total hence n=2)
  tx(1,1)=0.0
  ty(1,1)=0.0
  tx(1,2)=0.0 !Vxt
  ty(1,2)=0.0 !Vyt
  ionornot(1)=ionss/100
  tx(2,1)=tx(1,1)+d0*cos(dalpha/rad)
  ty(2,1)=ty(1,1)+d0*sin(dalpha/rad)
  tx(2,2)=0.0 !Vxt
  ty(2,2)=0.0 !Vyt
  ionornot(2)=ionss/100
end if
if(n.gt.2.and.aorm.eq.2)then
  D=D/sqrt(2.0)
  Dy=D
  call positions(nout,0,n/2,xneutral,yneutral,dmin,D,Dy)
  ionornot = 0
  do i=1,n/2
    alphas(i)= 0.0 !float(i)*4.0*pi/n
    tx(i,1)=xneutral(i)
    ty(i,1)=yneutral(i)
    tx(i+n/2,1)=xneutral(i)+d0*cos(alphas(i))
    ty(i+n/2,1)=yneutral(i)+d0*sin(alphas(i))
  end do
end if
if(n.gt.2.and.aorm.eq.1)then
  if(ions.gt.0.and.ions.lt.100)then
    di= sqrt(D*Dy*100.0/(pi*float(n)*float(ions)))
    call positions(nout,0,n,x,y,dmin,D,Dy)
    call distances(n,x0,y0,dd,x,y)
    call sorting3(n,dd,x,y)
    do i=1,n

tx(i,l)=x(i)
ty(i,l)=y(i)
ionornot(i)=2
end do
ionornot(1)=1
x0=x(1)
y0=y(1)
i1=1
do j=1,n*ionss/100
do i=1,n
if (ionornot(i).eq.2) then
  if (sqrt((x0-x(i))**2+(y0-y(i))**2).le.1.5*di) then
    ionornot(i)=0
  end if
end if
end do
call distances(n,x0,y0,dd,x,y)
dn=1000.0
do i=1,n
if (dd(i).lt.dm.and.ionornot(i).eq.2) then
  dm=dd(i)
i1=i
end if
end do
if (dm.eq.1000.0) then
  write(*,'(A,3I1)')'cannot place a new ion',j,i,exit
end if
ionornot(ii)=1
x0=x(ii)
y0=y(ii)
end do
do i=1,n
if (ionornot(i).eq.2) then
  ionornot(i)=0
end if
end do
end if

if(ionss.eq.0 .and. n.gt.2 .and. aom.eq.1) then
    di=0.0
    dmin=0.0

    call positions(nout,0,n*(100-ionss)/100,xneutral,yneutral,di, D, Dy)

    do i=1,n
        tx(i,1)=xneutral(i)
        ty(i,1)=yneutral(i)
        ionornot(i)=0
    end do

end if

if(ionss.eq.100 .and. n.gt.2 .and. aom.eq.1) then
    di=dmin
    d0=0.0

    call positions(nout,1,n*ionss/100,xion,yion,di, D, Dy)

    do i=1,n
        tx(i,1)=xion(i)
        ty(i,1)=yion(i)
        ionornot(i)=1
    end do

end if

end if

do i=1,n
    write(nout,401)i,ionornot(i),tx(i,1),ty(i,1)
end do

400 format (3x,'i',5x,'ion',5x,'x',5x,'y')
401 format(2i5,2f12.6)

end subroutine thetargets

subroutine rk4n(fcn,tin, tfin, ni, mi, Z1, n2, m2, Z2, f, xi, xf, ions, neq)

Solution for a system of n first-order ODEs
Method: Runge-Kutta 4th-order
Comment: can be easily used for n/2 second order ODEs
Alex G. February 2010, modified by Ana Samolov

use center

implicit none
integer neq,i,j,n1,n2,n1,ions(nt)
double precision tin, tfin
double precision f(neq*n2), xi(neq*n1), xf(neq*n1)
double precision h, t, m1, m2, Z1, Z2
double precision x(neq*n1), dx(neq*n1)
double precision kl(neq*n1),k2(neq*n1),k3(neq*n1),k4(neq*n1)

  h = tfin-tin
  t = tin

  x=0.0
  dx=0.0

  k1=0.0
  k2=0.0
  k3=0.0
  k4=0.0

  *= evaluate k1
  call fcn(t,nl,m1,Z1,n2,m2,Z2, f, xi, dx, ions, neq)
  do j=1,neq*n1
    k1(j) = h*dx(j)
    x(j) = xi(j) + k1(j)/2.0
  end do

  *= evaluate k2
  call fcn(t+h/2.0,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do J=1,neq*nl
    k2(J) = h*dx(j)
    x(J) = xi(J) + k2(J)/2.0
  end do

  *= evaluate k3
  call fcn(t+h/2.0,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do j=1,neq*n1
    k3(J) = h*dx(j)
    x(j) = xi(j) + k3(J)
  end do

  *= evaluate k4 and the result
  call fcn(t+h,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do j=1,neq*n1
    k4(J) = h*dx(j)
    xf(J) = xi(J) + k1(j)/6.0+k2(J)/3.0+k3(J)/3.0+k4(J)/6.0
  end do

end subroutine rk4n

subroutine dfcn(t,nl,m1,Z1,n2,m2,Z2,f,yin,yout,ions,neq)

  *= supports the runge-kutta subroutine with the needed equations of motion
  *= written by: Ana Semelov

  ! input ...
  ! t - simulation time
  ! n1 - number of given particles
  ! m1 - mass of given particles
  ! Z1 - charge of given particles
  ! yin - array of data of the particle positions and velocities, dimension n1*neq
  ! n2 - number of particles producing the potential we want to evaluate
  ! m2 - mass of particles producing the potential we want to evaluate
  ! Z2 - charge of particles producing the potential we want to evaluate

  ! *= evaluate kl
  call fcn(t,nl,m1,Z1,n2,m2,Z2, f, xi, dx, ions, neq)
  do j=1,neq*n1
    k1(j) = h*dx(j)
    x(j) = xi(j) + k1(j)/2.0
  end do

  *= evaluate k2
  call fcn(t+h/2.0,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do J=1,neq*nl
    k2(J) = h*dx(j)
    x(J) = xi(J) + k2(J)/2.0
  end do

  *= evaluate k3
  call fcn(t+h/2.0,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do j=1,neq*n1
    k3(J) = h*dx(j)
    x(j) = xi(j) + k3(J)
  end do

  *= evaluate k4 and the result
  call fcn(t+h,nl,m1,Z1,n2,m2,Z2, f, x, dx, ions, neq)
  do j=1,neq*n1
    k4(J) = h*dx(j)
    xf(J) = xi(J) + k1(j)/6.0+k2(J)/3.0+k3(J)/3.0+k4(J)/6.0
  end do

end subroutine dfcn
! f - array of data of other particles positions and velocities, dimension n2*neq
! ions - array of data whether the target particle is neutral or ion
! output ...
! yout - equations of motion of the tracked particle, dimension nl*neq

use center

implicit none

integer i, ik, neq, ions(nt), ni, n2, nl
double precision t, f(neq*n2), yin(neq*nl), yout(neq*nl)
double precision fx(nl), fy(nl), ml, m2, Zl, Z2
fx=0.0
fy=0.0

call forces(nl, ml, Zl, n2, m2, Z2, yin, f, fx, fy, ions, neq)

if(ml.le.m2) then
    ni=ni
end if

if(ml.gt.m2) then
    ni=centers
end if

do i=1, nl
    ik=(neq-1)*(i-1)+i
    yout(ik) = yin(ik+2)
    yout(ik+1) = yin(ik+3)
    yout(ik+2) = fx(i)
    yout(ik+3) = fy(i)
do end subroutine dfcn

subroutine forces(nl, ml, Zl, n2, m2, Z2, f1, f2, fx, fy, ions, neq)

Evaluate the potential in which a given particle moves
written by: Ana Samolov

! input ...
! nl - number of given particles
! ml - mass of given particles
! Zl - charge of given particles
! f1 - array of data of given particles positions and velocities, dimension nl*neq
! n2 - number of particles producing the force we want to evaluate
! m2 - mass of particles producing the force we want to evaluate
! Z2 - charge of particles producing the force we want to evaluate
! f2 - array of data of source particles positions and velocities, dimension n2*neq
! ions - array of data whether the target particle is neutral or ion
!
! output ...
! fx & fy - array of evaluated force components in
! the direction of motion and transverse direction

end subroutine forces
use center

implicit none

integer neq, ions(nt), n1, n2, i, j, ik, jk, ni, nii, a
double precision f1(ni*neq), f2(n2*neq)
double precision fx(ni), fy(ni)
double precision r, e, ml, m2, Zl, Z2
double precision d(nt)

fx = 0.0
fy = 0.0

! force on projectile (n1, m1, Z1 - index 1 for projectile)
if (m1.le.m2) then
  call distances(n1, nt, neq, f1, f2, d)
  call sortingI(neq, nt, d, f1, ions)
  ni = ni - centers
end if

! forces on a target (n1, m1, Z1 - index 1 for target)
if (m1.gt.m2) then
  call distances(n2, nt, neq, f2, f1, d)
  call sortingI(neq, nt, d, f2, ions)
  nii = nii + centers
end if

do i = 1, ni
  ik = (neq - 1) * (i - 1) + 1
  do j = 1, nii ! change
    jk = (neq - 1) * (j - 1) + j
    r = sqrt((f1(ik) - f2(jk))**2.0 + (f1(ik + 1) - f2(jk + 1))**2.0)
    if (ions(j) .eq. 0) then
      e = exp(-2.0*r)
      fx(i) = fx(i) + (2.0*r**2 + 2.0*r + 1) * (f1(ik) - f2(jk)) * e / (ml*r**3)
      fy(i) = fy(i) + (2.0*r**2 + 2.0*r + 1) * (f1(ik + 1) - f2(jk + 1)) * e / (ml*r**3)
    else
      fx(i) = f1(ik) + (Zl*Z2) * (f1(ik) - f2(jk)) / (ml*r**3) ! mu* on Coulomb
      fy(i) = f2(ik) + (Zl*Z2) * (f1(ik + 1) - f2(jk + 1)) / (ml*r**3)
    end if
  end do
end if

if (m2.gt.ml) then ! change
  do J = 1, nl
    if (i.ne.j) then
      jk = (neq - 1) * (j - 1) + j
      r = sqrt((f1(ik) - f1(jk))**2.0 + (f1(ik + 1) - f1(jk + 1))**2.0)
      fx(i) = fx(i) + (f1(ik) - f1(jk)) / (ml*r**3)
      fy(i) = fy(i) + (f1(ik + 1) - f1(jk + 1)) / (ml*r**3)
    end if
  end do
end if
end if
end do
end if
end do
if(ml.gt.m2.and.test.eq.l) then ! testing on rutherford scattering
fx=0.0
fy=0.0
end if
end subroutine forces

subroutine L0FC(nl,ml,Z1,n2,m2,Z2,f1,f2,Tk,V,Lz,ions,neq)
! calculates scattering angle
! and total energy and angular momentum in z-direction of the system
use center
implicit none
integer ni, nl, n2, neq, i, ik, ions(nt)
double precision fl(neq*nl),f2(neq*n2),Tk(nl),V(nl),Lz(nl)
double precision ml,m2, Zl, Z2
call potential(nl,ml,Zl,n2,m2,Z2,f1,f2,V,ions,neq)
if(ml.le.m2) then
ni=nl
end if
if(ml.gt.m2) then
ni=centers
end if
do i=1,ni
ik=(neq-1)+(i-1)*i
Tk(i)= 0.5*ml*(f1(ik+2)**2+f1(ik+3)**2)
Lz(i)= f1(ik)*f1(ik+3)-f1(ik+1)*f1(ik+2)
end do
end subroutine L0FC

subroutine potential(nl,ml,Z1,n2,m2,Z2,f1,f2,V,ions,neq)
! Evaluates the potential in which a given particle moves
! written by: Ana Samolov
! input ...
! ni - number of given particles
! ml - mass of given particles
! Z1 - charge of given particles
! f1 - array of data of given particles positions and velocities, dimension ni*neq
! n2 - number of particles producing the potential we want to evaluate
! m2 - mass of particles producing the potential we want to evaluate
! Z2 - charge of particles producing the potential we want to evaluate
! f2 - array of data of source particles positions and velocities, dimension n2*neq
! ions - array of data whether the target particle is neutral or ion
!
! output ...
! V - evaluated potential

use center

implicit none

integer ni, nii, ni, n2, neq, i, ik, j, jk
double precision f1(ni*neq),f2(n2*neq),V(ni)
double precision r, e, m1,m2,Z1,Z2,d(nt)
integer ions(nt)

V=0.0

if (m1.le.m2) then ! force on projectile (ni,m1,Z1- index 1 for projectile)
call distancesl(ni,nt,neq,f1,f2,d)
call sortingl(neq,nt,d,f2,ions)
i=m1
nii=centers
end if

if (m1.gt.m2) then ! forces on a target (ni,m1,Z1- index 1 for target)
call distancesl(n2,nt,neq,f2,f1,d)
call sortingl(neq,nt,d,f1,ions)
i=centers
nii=n2
end if

doi=1,ni

ik=(neq-1)*(i-1)+1

do j=1,nii ! change

jk=(neq-1)*(j-1)+j

r=sqrt((f1(ik)-f2(jk))**2.0+(f1(ik+1)-f2(jk+1))**2.0)

if(ions(j).eq.0) then
 e=exp(-2.0*r)
 V(i)=V(i)-(i+r)*e/r
else
 V(i) = V(i) + (Z1*Z2)/r ! mu+ on Coulomb
end if
end do

if (m1.le.m2) then

do j=1,ni

if(j.ne.i) then

jk=(neq-1)*(j-1)+j

end if
end do

if (m1.le.m2) then

do j=1,ni

if(j.ne.i) then

jk=(neq-1)*(j-1)+j

end if
end do
\[ r = \sqrt{((f1(ik) - f1(jk))^2 + (f1(ik+l) - f1(jk+l))^2)} \]

\[ V(i) = V(i) + \frac{(Z1*Z1)}{r} \]

end if

end do

end if

end do

end subroutine potential

end subroutine sorting2(n,xd,yd)

! Sorts the arrays in descending order and shuffles the corresponding data
! at the same time
! written by: Ana Samolov
!
! same as sorting just customise for specific data types
!

implicit none

integer i,n, swap
double precision xd(n),yd(n)
double precision tempx,tempy

swap=1

do while (swap.eq.1)

swap=0

do i=1,n-1
  if(xd(i).gt.xd(i+1)) then
    tempx=xd(i+1)
    xd(i+1)=xd(i)
    xd(i)=tempx
    tempy=yd(i+1)
    yd(i+1)=yd(i)
    yd(i)=tempy
    swap=1
  end if
  end do
  end do

end subroutine sorting2
 implicit none

 integer i,j,ik,iik,neq,npar, swap
 integer tempi, ions(npar)
 double precision xd(neq*npar),d(npar)
 double precision tempd,tempx(neq)

 swap=1
 do while (swap.eq.1)

 swap=0
 do i=1,npar-1

 ik=(neq-1)*i+(i-1)+1
 iik=(neq-1)*i+(i+1)

 if(d(i).gt.d(i+1)) then

 tempd=d(i+1)
 d(i+1)=d(i)
 d(i)=tempd

 tempi=ions(i+1)
 ions(i+1)=ions(i)
 ions(i)=tempi

 do j=1,neq

 tempx(j)=xd(iik+(j-1))
 xd(iik+(j-1))=xd(ik+(j-1))
 xd(ik+(j-1))=tempx(j)

 end do

 swap=1
 end if
 end do

 end do

 end subroutine sorting1

 !===============================================================================
 subroutine sorting(n,d,xd,yd)
 !===============================================================================
 ! Sorts the arrays in descending order and shuffles the corresponding data
 ! at the same time
 ! written by: Ana Samolov
 !===============================================================================
 ! input ...
 ! n - size of the arrays
 ! d() - data needed to be sort
 ! xd() & yd() - the corresponding arrays
 !===============================================================================
 implicit none

 integer i,n, swap, tempion
 double precision d(n),xd(n,2),yd(n,2)
 double precision tempd,tempx1,tempy1,tempx2,tempy2
swap = 1

do while (swap.eq.1)
    swap = 0
    do i = 1, n-1
        if (d(i).gt.d(i+1)) then
            tempd = d(i+1)
            d(i+1) = d(i)
            d(i) = tempd
            tempx1 = xd(i+1,1)
            xd(i+1,1) = xd(i,1)
            xd(i,1) = tempx1
            tempy1 = yd(i+1,1)
            yd(i+1,1) = yd(i,1)
            yd(i,1) = tempy1
            tempx2 = xd(i+1,2)
            xd(i+1,2) = xd(i,2)
            xd(i,2) = tempx2
            tempy2 = yd(i+1,2)
            yd(i+1,2) = yd(i,2)
            yd(i,2) = tempy2
            swap = 1
        end if
    end do
end do

end subroutine sorting

subroutine sorting3(n,d,xd,yd)

Sorts the arrays in descending order and shuffles the corresponding data at the same time
written by: Ana Samolov

input ...
! n - size of the arrays
! d() - data needed to be sort
! xd() & yd() - the corresponding arrays

implicit none

integer i,n, swap, tempion
double precision d(n), xd(n), yd(n)
double precision tempd, tempx1, tempy1

swap = 1

do while (swap.eq.1)
    swap = 0
    do i = 1, n-1
if (d(i).gt.d(i+1)) then
    tempd=d(i+1)
    d(i+1)=d(i)
    d(i)=tempd
    tempx=x(i+1)
    x(i+1)=x(i)
    x(i)=tempx
    tempy=y(i+1)
    y(i+1)=y(i)
    y(i)=tempy
    swap=1
end if
end do
end do
end subroutine sorting3

-----------------------------------------------------------------------
subroutine distances1(np,nt,neq,xp,xt,d)
! Evaluate the distances between the projectile and target particles
! written by: Ana Samolov
!-----------------------------------------------------------------------
! input ...
! np=neq - size of the arrays xp()
! nt=neq - size of the arrays xt()
! xp() - the array of positions of the projectile particles
! xt() - the arrays of positions of the target particles
! output ...
! d - array of evaluated values
!-----------------------------------------------------------------------
implicit none

integer i,ik,np,nt,neq
double precision xp(np*neq),x,y
double precision d(nt),xt(nt*neq)

x=0.0
y=0.0

do i=1,np
    ik=(neq-1)*(i-1)+1
    x=x+xp(ik)
    y=y+xp(ik+1)
end do

x=x/np ! x center of mass of projectiles
y=y/np ! y center of mass

do i=1,nt
    ik=(neq-1)*(i-1)+1
    d(i) = sqrt((x-xt(ik))**2.0+(y-xt(ik+1))**2.0)
end do
end subroutine distances

subroutine distances(n, x, y, d, xd, yd)

! Evaluate the distances between the target particles
! written by: Ana Samolov

! input ...
! n - size of the arrays xd(,), and yd(,)
! xd() - the array of x-positions of the target particles
! yd() - the arrays of y-positions of the target particles
! x - the x-position of the particle
! y - the y-position of the particle
! output ...
! d - array of evaluated values

implicit none

integer i, n
double precision x, y
double precision d(n), xd(n, 2), yd(n, 2)

do i = 1, n
  d(i) = sqrt((x-xd(i, 1))^2.0 + (y-yd(i, 1))^2.0)
end do
end subroutine distances

function deriv3(xx, xi, yi, ni, m)

! Evaluate first- or second-order derivatives
! using three-point Lagrange interpolation
! written by: Alex Godunov (October 2009)

! input ...
! xx - the abscissa at which the interpolation is to be evaluated
! xi() - the arrays of data abscissas
! yi() - the arrays of data ordinates
! ni - size of the arrays xi() and yi()
! m - order of a derivative (1 or 2)
! output ...
! deriv3 - interpolated value

implicit none

integer, parameter :: n = 3
double precision deriv3, xx
integer ni, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
integer i, j, k, ix

! exit if too high-order derivative was needed,
if (m > 2) then
  deriv3 = 0.0
  return
end if

! if x is outside the xi(1)-xi(ni) interval set deriv3=0.0
if (xx < xi(1) .or. xx > xi(ni)) then
  deriv3 = 0.0
  return
end if
! a binary (bisectional) search to find i so that xi(i-1) < x < xi(i)
i = 1
j = ni
do while (j > i+1)
k = (i+j)/2
if (xx < xi(k)) then
  j = k
else
  i = k
end if
end do

! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...
i = 1 + 1 - n/2

! check boundaries: if i is outside of the range [1, ... n] -> shift i
if (i < 1) i=1
if (i + n > ni) i=ni-n+1

! old output to test i
write(*,100) xx, i
! 100 format (f0.5, i5)

! just wanted to use index i
ix = i

! initialization of f(n) and x(n)
do i=l,n
  f(i) = yi(ix+i-1)
  x(i) = xi(ix+i-1)
end do

! calculate the first-order derivative using Lagrange interpolation
if (m == 1) then
  deriv3 = (2.0*xx - (x(2)+x(3)))*f(1)/((x(1)-x(2))*(x(1)-x(3)))
  deriv3 = deriv3 + (2.0*xx - (x(1)+x(3)))*f(2)/((x(2)-x(1))*(x(2)-x(3)))
  deriv3 = deriv3 + (2.0*xx - (x(1)+x(2)))*f(3)/((x(3)-x(1))*(x(3)-x(2)))
else
  deriv3 = 2.0*f(1)/((x(1)-x(2))*(x(1)-x(3)))
  deriv3 = deriv3 + 2.0*f(2)/((x(2)-x(1))*(x(2)-x(3)))
  deriv3 = deriv3 + 2.0*f(3)/((x(3)-x(1))*(x(3)-x(2)))
end if

end function deriv3

! subroutine differentiation(x,y,dy,m)
! Evaluate first-order derivatives - routine to support any differentiation method
! written by: Ana Samolov
!
implict none
!
integer n, i, m
double precision x(n), y(n), dy(n), xx
double precision deriv3

do i=1,n
    xx=x(i)
    dy(i)=deriv3(xx, x, y, n, m)
end do

end subroutine differentiation

============================================================================
APPENDIX C

QUANTILES AND Q-Q PLOTS

When we need to compare the shapes of the two distributions, we can do that by comparing the two histograms. However, this may introduced the observer bias. The other more sensitive way is to use the quantile or q-q plots. If we have a set of \( n \) data points \( x_n \) that may be divided into \( q \) data subsets then the quantiles are the boundery values between the two subsets. In this sense a \( k^{th} \) quantile would be a data value below which \( k/q \) fraction of data values can be find. So 0.2 quantile is the data value below which 20% of data lies. One example of the quantile is the sample median. It is a 0.5 quantile and it is the data value that divides upper part of data distribution from the lower part. In other words it is a measure of the center of a distribution or the middle value of the ordered data. The q-q plots are then scatter plots of the quantiles of first data set against the same quantiles of the second data set. If the two data sets are identically distributed then then their q-q plot will have a straight line \( y = x \) [56]. If the two sets of data are lineally dependent then their q-q plot will still be linear but with changed location and slope. This implies that any departure from this linearity is the indication of the differences between two data sets. The possible interpretations of the q-q plots are as follows:

- all but a few points fall on a line - presence of outliers in the data (points numerically distant from rest of the points);
- left end of pattern is below the line while right end of pattern is above the line - long tails at both ends of the data distribution plotted on the abscissa ;
- left end of pattern is above the line while right end of pattern is below the line - short tails at both ends of the data distribution plotted on the abscissa ;
- curved pattern with slope increasing from left to right - data distribution on abscissa is skewed to the right;
- curved pattern with slope decreasing from left to right - data distribution on abscissa is skewed to the left
• staircase pattern (plateaus and gaps) - data have been rounded or are discrete;

For the case in which the data distributions have heavy tails the q-q plot tends to emphasize the comparative structure in the tails and to blur the distinctions in the middle where the densities are high. The reason for this is that the quantile function is a rapidly changing when the densities are low as it is in the tails and a slowly changing one in the middle where the densities are high.
APPENDIX D

DIRECT INTEGRATION CODE

============================================================================
module inputdata

integer, parameter :: nr=17
integer, parameter :: nt=45

double precision r(nr), ltheta(nr)
double precision thetas(nt), ltheta(nt)
double precision theta, p0

end module inputdata

============================================================================
module splining


end module splining

============================================================================
module direct

integer iss

end module direct

============================================================================
PROGRAM Radontransform

by Ana Samolov November 08, 2011

! Inverse Radon Transform for 2D Plasma Tomography
!
! Intensities on different positions and under different angles, I(r,theta)
!
! output...
! Distribution function g(x,y)

use inputdata
use splining

IMPLICIT NONE

double precision, parameter :: pi = 3.141592653
double precision, parameter :: rad = 57.2958

double precision, parameter :: Radius = 1.6 !radius of plasma and quarc tube
double precision, parameter :: thetamin = 2.0

! QUANC parameters
integer nofun

double precision errest, flag
double precision, parameter :: abserr=0.0, relerr=1.0e-6

! working variables

============================================================================
integer, parameter:: nx= 161 !17
integer, parameter:: ny= 161 !17
integer i, j, k, ix, iy, m, mm

double precision INtheta(nr*nt)
! real for qagse
double precision l1, l2, g, gxy(ny), gps(nx, ny), Agps(nx, ny)
double precision x, y, pomin, pomin, d11, logp

double precision dtheta
double precision, parameter:: dr=0.2
double precision dx
double precision dy
double precision, parameter:: dtmin=0.0
double precision deriv3

! for integration subroutine
integer, parameter :: limit=1000
real, parameter :: abserrs=0.0, relerrs=1.0e-6
real alist, blist, rlist, elist, epsabs
integer neval, ier, last, lord
dimension alist(limit), blist(limit), rlist(limit), elist(limit), iord(limit)

! direct integration
common/parameters/dx, dy

external fp, ftheta
double precision lspline, naxval, gauss16

! open(unit=9, file=’706datale-4quanc8.dat’)
! open(unit=9, file=’714datale-4quanc8.dat’)
open(unit=9, file=’smoothdatainv.dat’)
open(unit=12, file=’analytical inverse.dat’)

INtheta=0.0
r=0.0
lptheta=0.0
thetas=0.0
Itheta=0.0
gxy=0.0
dtheta = 176.0/(nt-1) ! 6.0
dx=2.0*Radius/float(nx-1)
dy=2.0*Radius/float(ny-1)
write(*,*) radius, dx, dy
! read(*,*) r

call reading(INtheta)
do ix=1, nx

write(*,*) ix
x=float(ix-1)*dx-Radius

do iy=1, ny
y=float(iy-1)*dx/RADIUS

if((x**2.0+y**2.0).le.RADIUS**2.0) then
  !write(*,*) x,y
  do i=1,nt
    lptheta=0.0
    r=0.0
    theta= (thetamin+dtheta*float(i-1))
    thetas(i)= theta/rad
    p0=-x*sin(thetas(i))+y*cos(thetas(i))
    k=nr*(i-1)
    do j=1,nr
      lptheta(j)=Ithetas(k+j)
      r(j)=-1.0*(float(j-1)*dr-RADIUS)
    end do
    call sorting2(nr,r,lptheta)
    call spline (r, lptheta, b, c, d, nr)
    p0min= 1E-6 !-Radius
    p0max=RADIUS
    call quanc8(fp,p0min,RADIUS,abser,relerr,l1,errest,nofun,flag)
    Ithetas(i)=l1
    end do
  call spline (thetas, Ithetas, b, c, d, nt)
  call quanc8(ftheta,dtoin,pi,abser,relerr,g,errest,nofun,flag)
else
  g=0.0
end if

  gxy(iy)=g/(2*pi**2)
  Agps(ix,iy)=exp(-(x**2+y**2))
end do

write(9,200)(gxy(iy),iy=1,ny)
write(12,200)(Agps(ix,iy),iy=1,ny)
end do

!direct integration

  call directK(gxy, Radius, thetamin, dtheta, dr, nx, ny)

  stop

200 format(161(e15.4))
END PROGRAM Radontransform

double precision function fp(p)
use inputdata
use splining
implicit none
integer i
double precision p
!double precision b(nr),c(nr),d(nr)

double precision ispline,deriv3,deriv4

!call spline (r, Iptheta, b, c, d, nr+2)
!fp=deriv3(p,p0, r, Iptheta, nr, 1)-deriv3(p0-p, r, Iptheta, nr, 1)
!fp=ispline(p0-p, r, Iptheta, b, c, d, nr)+ispline(p*p0, r, Iptheta, b, c, d, nr)-
- 2* ispline(p0, r, Iptheta, b, c, d, nr)
fp=deriv4(p,p0, r, Iptheta, nr, 3, 1)-deriv4(p0-p, r, Iptheta, nr, 3, 1)
!fp=ispline(p0-p, r, Iptheta, b, c, d, nr)+ispline(p*p0, r, Iptheta, b, c, d, nr)
!fp = ispline(p, r, Iptheta, b, c, d, nr)
fp=fp/ (p-p0)**2

end function fp

use inputdata
use splining
implicit none
double precision thetai ,b(nt),c(nt),d(nt)

double precision ispline

!call spline (thetas, Itheta, b, c, d, nt)
!theta=ispline(thetal, thetas, Itheta, b, c, d, nt)

end function ftheta

use direct
use splining
implicit none
integer i
integer i
double precision s

double precision ispline,deriv3,deriv4

!fs=deriv3(s, ss, Is, iss, 1)
fs=ispline(s, ss, Is, b, c, d, iss)
!fp=deriv4(s, ss, Is, iss, 3, 1)

end function fs

subroutine reading(intheta)
! reading population data from a file and rewriting it to one array \( \theta \)

```fortran
use inputdata

implicit none

integer i,j,k,kk
double precision intheta(nr*nt),I0(nr,nt)

open(unit=7, file='smoothdata.dat')
!open(unit=7, file='smoothdata714.dat') !706inputdata
!open(unit=7, file='714inputdata.dat')
open(unit=8, file='testread.dat')

IO=0.0

do i=1,nr
    read (7,*) (I0(i,j),j=1,nt)
    write (8,100) (I0(i,j),j=1,nt)
end do

do i=1,nt
    do j=1,nr
        k=nr*(i-1)+j
        intheta(k)=I0(j,i)
    end do
end do

100 format (21(fl0.4))
end subroutine reading
```

subroutine quanc8(fun,a,b,abserr,relerr,results,errest,nofun,flag)

! estimate the integral of \( f(x) \) from \( a \) to \( b \)
! to a user provided tolerance.
! an automatic adaptive routine based on
! the 8-panel newton-cotes rule.

input ..
!
fun the name of the integrand function subprogram \( f(x) \).
!
a the lower limit of integration.
!
b the upper limit of integration. (b may be less than a.)
!
relerr a relative error tolerance. (should be non-negative)
!
abserr an absolute error tolerance. (should be non-negative)
!
output ..
!
result an approximation to the integral hopefully satisfying the
! least stringent of the two error tolerances.
!
errest an estimate of the magnitude of the actual error.
!
nofun the number of function values used in calculation of result.
!
flag a reliability indicator. if flag is zero, then result
! probably satisfies the error tolerance. if flag is
! xxx.yyy , then xxx = the number of intervals which have
! not converged and 0.yyy = the fraction of the interval
! left to do when the limit on nofun was approached.
double precision fun, a, b, abserr, relerr, results, errest, flag
integer nofun

double precision w0,w1,w2,w3,area,x0,f0,stone,step,cor1, temp
double precision qprev,qnov,qdiff,qleft,esterr,tolerr
double precision qright(31),f(16),x(16),fsave(8,30),xsave(8,30)
double precision dabs, dmaxl

integer levmin, levmax, levout, nomax, nofin, lev, nim, i, j

*** stage 1 *** general initialization

set constants.

levmin = 1
levmax = 30
levout = 6
nomax = 5000
nofin = nomax - 8 * (levmax - levout + 2 * (levout + 1))

trouble when nofun reaches nofin

w0 = 3956.0d0 / 14175.0d0
w1 = 23552.0d0 / 14175.0d0
w2 = -3712.0d0 / 14175.0d0
w3 = 41984.0d0 / 14175.0d0
w4 = -18160.0d0 / 14175.0d0

initialize running sums to zero.

flag = 0.0d0
results = 0.0d0
cor1 = 0.0d0
errest = 0.0d0
area = 0.0d0
nofun = 0

if (a .eq. b) return

*** stage 2 *** initialization for first interval

lev = 0
nim = 1
x0 = a
x(16) = b
qprev = 0.0d0
f0 = fun(x0)

stone = (b - a) / 16.0d0
x(0) = (x0 + x(16)) / 2.0d0
x(4) = (x0 + x(8)) / 2.0d0
x(12) = (x(8) + x(16)) / 2.0d0
x(2) = (x0 + x(4)) / 2.0d0
x(6) = (x(4) + x(8)) / 2.0d0
x(10) = (x(8) + x(12)) / 2.0d0
x(14) = (x(12) + x(16)) / 2.0d0

do 25 j = 2, 16, 2
   f(j) = fun(x(j))
25 continue

nofun = 9

*** stage 3 *** central calculation

requires qprev,x0,x2,x4,...,x16,f0,f2,f4,...,f16.

calculates x1,x3,...,x15, f1,f3,...,f15,qleft,qright,qnov,qdiff,area.

30 x(1) = (x0 + x(2)) / 2.0d0
f(1) = fun(x(1))
do 35 j = 3, 16, 2
\( x(j) = \frac{x(j-1) + x(j+1)}{2.040} \)

\( f(j) = \text{fun}(x(j)) \)

35 continue

\[ \text{nofun} = \text{nofun} + 8 \]

\[ \text{step} = \frac{(x(16) - x_0)}{16.040} \]

\[ \text{qleft}(\text{lev}+1) = (w_0*(f(8)+f(16)) + v_1*(f(9)+f(15))) + w_3*(f(10)+f(14)) + w_3*(f(11)+f(13)) + w_4*f(12) * \text{step} \]

\[ \text{qright}(\text{lev}+1) = (w_0*(f(8)+f(16)) + v_1*(f(9)+f(15))) + w_3*(f(10)+f(14)) + w_3*(f(11)+f(13)) + w_4*f(12) * \text{step} \]

\[ \text{qnov} = \text{qleft} + \text{qright}(\text{lev}+1) \]

\[ \text{qdiff} = \text{qnov} - \text{qprev} \]

\[ \text{area} = \text{area} + \text{qdiff} \]

\* *** stage 4 *** interval convergence test \*

\[ \text{esterr} = \frac{\text{dabs}(\text{qdiff})}{1023.040} \]

\[ \text{tolerr} = \frac{\text{dmax}(\text{dabs,relerr}+\text{dabs}(\text{area}))}{\text{step}/\text{stone}} \]

if (\( \text{lev} \lt \text{levmin} \)) go to 50

if (\( \text{lev} \geq \text{levmax} \)) go to 62

if (\( \text{nofun} > \text{noin} \)) go to 60

if (\( \text{esterr} \leq \text{tolerr} \)) go to 70

\* *** stage 5 *** no convergence \*

locate next interval.

50 \( \text{nmin} = 2*\text{nmin} \)

\( \text{lev} = \text{lev}+1 \)

store right hand elements for future use.

\[ \text{do 52 } i = 1, 8 \]

\[ \text{fsave}(i,\text{lev}) = f(i+8) \]

\[ \text{xsave}(i,\text{lev}) = x(i+8) \]

52 continue

assemble left hand elements for immediate use.

\[ \text{qprev} = \text{qleft} \]

\[ \text{do 55 } i = 1, 8 \]

\[ j = -1 \]

\[ f(2*j+18) = f(j+9) \]

\[ x(2*j+18) = x(j+9) \]

55 continue

go to 30

\* *** stage 6 *** trouble section \*

number of function values is about to exceed limit.

60 \( \text{nofin} = 2*\text{nofin} \)

\( \text{levmax} = \text{levout} \)

\( \text{flag} = \text{flag} + (b - x_0) / (b - a) \)

go to 70

current level is levmax.

62 \( \text{flag} = \text{flag} + 1.040 \)

\* *** stage 7 *** interval converged \*

add contributions into running sums.

70 \( \text{results} = \text{results} + \text{qnov} \)

\( \text{errest} = \text{errest} + \text{esterr} \)

\( \text{corll} = \text{corll} + \text{qdiff} / 1023.040 \)

locate next interval.

72 if (\( \text{nmin} \leq 2*\text{nmin}/2 \)) go to 75
nim = nim/2
lev = lev-1
go to 72
75 nim = nim + 1
    if (lev .le. 0) go to 80

!! assemble elements required for the next interval.

qprev = qright(lev)
x0 = x(1)
f0 = f(1)
do 78 i = 1, 8
   f(2*i) = fsave(1, lev)
   x(2*i) = xsave(1, lev)
78 continue

!! *** stage 8 *** finalize and return

results = results + cori1

! make sure errest not less than roundoff level.

if (errest .eq. 0.0d0) return
82 temp = dabs(results) + errest
    if (temp .ne. dabs(results)) return
    errest = 2.0d0*errest
go to 82
end

===============================================

subroutine spline (x, y, b, c, d, n)

Calculate the coefficients b(i), c(i), and d(i), i=1,2,...,n
for cubic spline interpolation
s(x) = y(i) + b(i)*(x-x(i)) + c(i)*(x-x(i))^2 + d(i)*(x-x(i))^3
for x(i) < x < x(i+1)

Alex G: January 2010

--------------------------------------------------------------------------------------------------

implicit none
integer n
double precision x(n), y(n), b(n), c(n), d(n)
integer i, j, gap
double precision h

gap = n-1
! check input
if (n < 2 ) return
if (n < 3 ) then
   b(1) = (y(2)-y(1))/(x(2)-x(1)) ! linear interpolation
   c(1) = 0.
   d(1) = 0.
   b(2) = b(1)
   c(2) = 0.
   d(2) = 0.
return
end if

! step 1: preparation
!
d(1) = x(2) - x(1)
c(2) = (y(2) - y(1))/d(1)
do i = 2, gap
d(i) = x(i+1) - x(i)
b(i) = 2.0*(d(i-1) + d(i))
c(i+1) = (y(i+1) - y(i))/d(i)
c(i) = c(i+1) - c(i)
end do
!
! step 2: end conditions
!
b(1) = -d(1)
b(n) = -d(n-1)
c(1) = 0.0
c(n) = 0.0
if(n /= 3) then
  c(i) = (c(3)/(x(4)-x(2)) - c(2)/(x(3)-x(1))
c(n) = c(n-1)/(x(n)-x(n-2)) - c(n-2)/(x(n-1)-x(n-3))
c(i) = c(i)*d(i)**2/(x(4)-x(i))
c(n) = -c(n)*d(n-1)**2/(x(n)-x(n-3))
end if
!
! step 3: forward elimination
!
do i = 2, n
  h = d(i-1)/b(i-1)
b(i) = b(i) - h*d(i-1)
c(i) = c(i) - h*c(i-1)
end do
!
! step 4: back substitution
!
c(n) = c(n)/b(n)
do j = 1, gap
  i = n-j
  c(i) = (c(i) - d(i)*c(i+1))/b(i)
end do
!
! step 5: compute spline coefficients
!
b(n) = (y(n) - y(gap))/d(gap) + d(gap)*(c(gap) + 2.0*c(n))
do i = 1, gap
  b(i) = (y(i+1) - y(i))/d(i) - d(i)*(c(i+1) + 2.0*c(i))
d(i) = (c(i+1) - c(i))/d(i)
c(i) = 3.0*c(i)
end do
c(n) = 3.0*c(n)
d(n) = d(n-1)
end subroutine spline

function ispline(u, x, y, b, c, d, n)
! function ispline evaluates the cubic spline interpolation at point z
! ispline = y(i)+b(i)*(u-x(i))+c(i)*(u-x(i))^2+d(i)*(u-x(i))^3
! where x(i) <= u < x(i+1)
!
! input...
! u = the abscissa at which the spline is to be evaluated
! x, y = the arrays of given data points
! b, c, d = arrays of spline coefficients computed by spline
! n = the number of data points
! output:
! ispline = interpolated value at point u
end function ispline
implicit none
double precision ispline
integer n
double precision u, x(n), y(n), b(n), c(n), d(n)
integer i, j, k
double precision dx

! if u is outside the x() interval take a boundary value (left or right)
if(u <= x(1)) then
   ispline = y(1)
   return
end if
if(u >= x(n)) then
   ispline = y(n)
   return
end if
!
! binary search for i, such that x(i) <= u <= x(i+1)
!
i = 1
j = n+1
do while (j > i+1)
   k = (i+j)/2
   if(u < x(k)) then
      i = k
   else
      j = k
   end if
end do
!
! evaluate spline interpolation
!
   dx = u - x(i)
   ispline = y(i) + dx*(b(i) + dx*(c(i) + dx*d(i))
end function ispline

function deriv3(xx, xi, yi, ni, m)
! Evaluate first- or second-order derivatives
! using three-point Lagrange interpolation
! written by: Alex Godunov (October 2009)

implicit none
integer, parameter :: n=5
double precision deriv3, xx
integer ni, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
integer i, j, k, ix

! exit if too high-order derivative was needed,
if (m > 2) then
   deriv3 = 0.0
   return
end function deriv3
if x is outside the \( x(i-1) - x(i) \) interval set deriv3=0.0
\[ \text{if } (x < x(i) \text{ or } x > x(i)) \text{ then} \]
\[ \text{deriv3} = 0.0 \]
return
end if

! a binary (bisectional) search to find i so that \( x(i-1) < x < x(i) \)
i = 1
j = ni
do while (j > i+1)
k = (i+j)/2
if (xx < xi(k)) then
    j = k
else
    i = k
end if
end do

! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in \( x_{i-1}, x_{i+1}, x_{i+2} \ldots \)
i = i + 1 - n/2

! check boundaries: if i is outside of the range \([1, \ldots n]\) -> shift i
\[ \text{if } (i < 1) \text{ i=1} \]
\[ \text{if } (i + n > ni) \text{ i=ni-n+1} \]

! old output to test i
! write(*,100) xx, i
! 100 format (f10.5, I5)

! just wanted to use index i in x
ix = i

! initialization of \( f(n) \) and \( x(n) \)
do i=1,n
f(i) = yi(ix+i-1)
x(i) = xi(ix+i-1)
end do

! calculate the first-order derivative using Lagrange interpolation
if (m == 1) then
    deriv3 = \[ (2.0*xx - (x(2)+x(3)))*(f(1)/(x(1)-x(2)))*(x(1)-x(3)) \]
    deriv3 = deriv3 + \[ (2.0*xx - (x(1)+x(3)))*(f(2)/(x(2)-x(1)))*(x(2)-x(3)) \]
    deriv3 = deriv3 + \[ (2.0*xx - (x(1)+x(2)))*(f(3)/(x(3)-x(1)))*(x(3)-x(2)) \]
end if
end function deriv3
! n - number of points to evaluate derivatives
! m - order of a derivative (1 or 2)
! output ...  
! deriv4 - the first- or second-order derivative

implicit none

double precision deriv4, xx
integer ni, n, m

double precision xi(ni), yi(ni)

double precision x(n), f(n)

double precision d1, d2, d3, h, s
integer 1, j, k, ix

! exit if too high-order derivative was needed,  
! or too many base points were needed for derivatives  
if (m > 2 .or. m = n .or. n > 5) then  
deriv4 = 0.0
return
end if

! if x is outside the xi(1)-xi(ni) interval set deriv4=0.0  
if (xx < xi(1) .or. xx > xi(ni)) then  
deriv4 = 0.0
return
end if

! a binary (bisectional) search to find i so that xi(i-1) < x < xi(i)  
i = 1
j = ni
do while (j > i+1)
    k = (i+j)/2
    if (xx < xi(k)) then  
        j = k
    else  
        i = k
    end if  
end do

! shift i that will correspond to n-th order of interpolation  
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...  
i = 1 + 1 - n/2

! check boundaries: if i is outside the range [1, ... n] -> shift i  
if (i < 1) i=1  
if (i + n > ni) i=ni-n+1

! old output to test i  
! write(*,100) xx, i  
! 100 format (f10.5, I5)

! just want to use index i  
ix = i

! initialization of f(n) and x(n)  
do i=1,n
    f(i) = yi(ix+i-1)
    x(i) = xi(ix+i-1)
end do

! calculate divided difference coefficients  
d1 = f(2) - f(1)  
if (n > 2) d2 = f(3) - 2.0*ef(2) + f(1)  
if (n > 3) d3 = f(4) - 3.0*ef(3) + 3.0*ef(2) - f(1)

h = x(2) - x(1)
s = (xx - x(1))/h

! calculate the first order derivative
if (m == 1) then
    deriv4 = (1.0/h)*d1
end if
if (n > 2) deriv4 = deriv4 - (1.0/h)*((2.0*a-1.0)/2.0)*d2
if (n > 3) deriv4 = deriv4 - (1.0/h)*((3.0*a-6.0*a+2.0)/6.0)*d3
end if

function deriv4

end function deriv4

subroutine gaussl6(f,a,b)

implicit none
integer, parameter :: n=8
double precision gaussl6, f, a, b
double precision ti(n), ci(n)
data ti/0.0950125098, 0.2816035507, 0.4580167776, 0.6178762444, 0.7554044083, 0.8656312023, 0.9445750230, 0.9894009349/
data ci/0.1894506104, 0.1826034150, 0.1691565193, 0.1495959888, 0.1246289712, 0.0951585116, 0.0622535239, 0.0271524594/
double precision r, m, c
integer 1
r = 0.0;
m = (b-a)/2.0;
c = (b+a)/2.0;
do i = 1,n
    r = r + ci(i)*(f(m*(-1.0)*ti(i) + c) + f(m*ti(i) + c))
end do
gauss16 = r*m
return
end function gauss16

SUBROUTINE Gauss24 (f,a,b,I)

implicit none
DOUBLE PRECISION a,b,f,I,c,m
DOUBLE PRECISION, dimension (12) ::
XI=/(0.0640568929,0.1911188675,0.3150426797,0.4337935076,
DOUBLE PRECISION, dimension (12) ::
Wl- (/0.1279381953, 0.1258374563, 0.1216704729, 0.1155056680,
0.107442701, 0.0976186521, 0.0861901615, 0.073464814,
0.0592985849, 0.0442774388, 0.0285313886, 0.0123412297/)
INTEGER*4 J
external f
m- (b-a)/2.0
c- (b+a)/2.0
DO j=1, 12, 1
I- I + Wi(j) * (f(m*(-1.0)*Xi(j)+c) + f(m*Xi(j)+c))
END DO
I- I = m
RETURN
END SUBROUTINE Gauss24

!-----------------------------------------------------------------------------
! subroutine directHgf, Radius, thetamln, dtheta, dr, nx, ny)
!-----------------------------------------------------------------------------
use inputdata
use splining
use direct
Implicit none
! common
integer nx, ny

double precision, parameter :: rad = 57.2958
double precision, parameter :: pi = 3.141592653
integer n
!real Radius for qagse

double precision Radius, thetamin, dtheta, dx, dy
common/parameters/dx, dy

integer nofun
double precision errstat, flag
double precision, parameter :: abserr=1.0e-6, relerr=1.0e-6

!working
integer i, j, it, ip, iis, ix, iy

double precision s(1000),IsA(1000), smax, smin, ds, Its

double precision x(1000), y(1000), xx(nx), yy(ny)

double precision gf(nx,ny), f(nr,nt), fA(nr,nt)

double precision delta, di, dj, dn, p

external fs
open(unit=10, file='direct.dat')
open(unit=11, file='analytical.dat')
n=nx

do i = 1, nx, 1
    xx(i) = dx*(i-1)-Radius
    yy(i) = dy*(i-1)-Radius
end do

dn=2*Radius/(n-1)

do iis = 1, n, 1
    x(iis) = dn*(iis-1)-Radius
end do

do it=1, nt, 1 !change from to 179, 1
    theta = thetanin+dtheta*(it-1) !it
    do ip=1, nr, 1 ! change from nr to 101
        p = -(dr*(ip-1)-Radius)!*sin(theta/rad) ! change from dr to 3.2/50
        do iis=1, n, 1
            y(iis) = x(iis)*tan(theta/rad)-p/cos(theta/rad)
            if(theta.eq.90) then
                y(iis) = p
            end if
        end do
        if(theta.eq.90 .and. abs(p).ge.1.6) then
            smax=0.0001
        else
            smax=(Radius**2-p**2)**0.5
        end if
        smin=smax
        !write(*,*) smin, smax
    end do
    ds=(2.0*Radius)/(n-1)
    iss=0
    s=0.0
    !write(10,100)theta, p, smin, smax
end do iis = 1, n, 1
\[ s(iis) = x(iis)\cos(\theta/\text{rad}) + y(iis)\sin(\theta/\text{rad}) \]

if (\theta \equiv 90) then
   \[ s(iis) = x(iis) \]
end if

!write(*,*) iis, s(iis), smin, smax

if (s(iis) \ge smin and s(iis) \le smax) then
   call binary(y(iis), yy, ny, i)
   !write(*,*) s(iis), y(iis), yy(i), i
   di = abs(y(iis) - yy(i)) ! left
   dii = abs(y(iis) - yy(i+1)) ! right
   if (di > dii) then
      j = i+1 ! closer to the right
      delta = dii
      !write(*,*) 'dii', delta, j
   else
      j = i ! closer to the left
      delta = di
      !write(*,*) 'di', delta, j
   end if
   if (abs(delta/yy(j)) \lt 1.E-1) then
      ix = iis
      iy = j
      iss = iss+1
      !write(*,*) 'entered sampling', 'iss=', iss, 'ix=', ix, 'iy=', iy
      Is(iss) = gf(ix, iy) ! \( Is(iis) = IsA(iis) \)
      ss(iss) = s(iis) ! \( ss(iis) = s(iis) \)
   end if
end if
end do

call sorting2(iss, ss, Is)
!do i = 1, iss
!write(10,102) i, ss(i), Is(i)
!end do
if(iss.gt.2) then
call spline (ss, Is, b, c, d, iss)
call quanc8(fs,amin,smax,aberr,relerr,Its,errest,nofun,flag)
else
Its=0.0
end if
f(ip,it)=Its
fA(ip,it)=(pi=0.5)*exp(-(p**2))
end do
end do
de ip=1,nr,1 ! change from nr to 100
write(10,100)(f(ip,it),it=1,nt) ! change from it=1, nt to it=1,179
write(11,100)(fA(ip,it),it=1,nt)
end do
100 format(161(e15.4)) ! change from 161 to 179
101 format (9x,'theta',10x, 'p',15x, 's',14x, 'z',14x, 'y', 5x,'ix',3x,
 'iy',2x, 'iss',5x, 'ss(iss)',8x, 'Is(iss)', 7x,'gf(ix,iy)
 ,/5(e15.4),3i4,3(e15.4),/)!
102 format(14,4(e15.4),2i4,2(e15.4))
end subroutine directI
!==============================================================================
subroutine binary(xx,xi,ni,i)
!==============================================================================
implicit none
integer ni,i,j,k
double precision xx,xi(ni)
! a binary (bisectional) search to find i so that xi(i) < x < xi(i+1)
i = 1
j = ni
do while (j > i+1)
k = (i+j)/2
if (xx < xi(k)) then
  j = k
else
  i = k
end if
end do
end subroutine
!==============================================================================
subroutine sorting2(xd,xd,yd)
!==============================================================================
! Sorts the arrays in descending order and shuffles the corresponding data
! at the same time
! written by: Ana Samolov
!---------------------------------------------------------------
! same as sorting just customise for specific data types
!---------------------------------------------------------------
implicit none
integer i,n, swap
double precision xd(n),yd(n)
double precision tempx,tempy

swap=1

do while (swap.eq.1)

swap=0

do i=1,n-1

if(xd(i).gt.xd(i+1)) then

tempx=xd(i+1)
xd(i+1)=xd(i)
xd(i)=tempx

tempy=yd(i+1)
yd(i+1)=yd(i)
yd(i)=tempy

swap=1

end if

end do

end do

end subroutine sorting2
APPENDIX E

FILTERED BACK PROJECTION CODE

module inputdata

integer, parameter :: nr = 17
integer, parameter :: nt = 21

double precision r(nr), Iptheta(nr)
double precision thetas(nt), Itheta(nt)
double precision theta, p0
end module inputdata

module splining

end module splining

module direct

integer iss
end module direct

PROGRAM Radontransform

! by Ana Samolov November 08, 2011
!
! Inverse Radon Transform for 2D Plasma Tomography
! input...
! Intensities on different positions and under different angles, I(r,theta)
! output...
! Distribution function g(x,y)

use inputdata
use splining

IMPLICIT NONE

double precision, parameter :: pi = 3.141592653
double precision, parameter :: rad = 57.2958

double precision, parameter :: Radius = 1.6 !radius of plasma and quarc tube
double precision, parameter :: thetamin = 48.0

! QUARC parameters
integer nefun

double precision errest, flag
double precision, parameter :: abserr=0.0, relerr=1.0e-6
! working variables

integer, parameter :: nx = 161 !17
integer, parameter :: ny = 161 !17
integer i, j, k, ix, iy, mm

double precision INtheta(nr*nt), Ptheta(nr,nt), Qtheta(nr,nt), h(nr), shepp(nr)
double precision Q(nr)
! real for qagse
double precision It1, It2, g, gxy(ny), gps(nx, ny), Agps(nx, ny)
double precision x, y, pomin, p0max, dll

double precision dtheta

double precision, parameter :: dr = 0.2
double precision dx
double precision dp, dn

double precision, parameter :: dtmin = 0.0

double precision deriv3

! for integration subroutine

integer, parameter :: limit = 1000
real, parameter :: abserrs = 0.0, relerrs = 1.0e-6
real aList, bList, rList, eList, epsabs
integer neval, ier, last, iord

dimension aList(limit), bList(limit), rList(limit), eList(limit), iord(limit)

! direct integration

common/parameters/dx, dy

external fp, ftheta
double precision ispline, maxval, gauss16

open(unit=9, file='714dataKamLak.dat')
!open(unit=9, file='714dataKam.dat')
open(unit=12, file='analytical inverse.dat')

dtheta = 6.0 !(176/(nt-1))

INtheta = 0.0
r = 0.0
Iptheta = 0.0
theta = 0.0
Itheta = 0.0
gxy = 0.0
dx = 2.0*Radius/float(nx-1)
dy = 2.0*Radius/float(ny-1)
dx = 2.0*Radius/float(nr-1)

write(*,*) radius, dx, dy
! read(*,*) r

! call reading(INtheta)

do ix = 1, nx
write(*,*) ix
x=float(ix-1)*dx+Radius

do iy=1,ny
y=float(iy-1)*dy+Radius
if((x**2.0+y**2.0).le.Radius**2.0) then
!write(*,*) x,y
Ptheta=0.0
Qtheta=0.0

do i=1,nt
Q=0.0
I*theta=0.0
r=0.0
theta= (thetamin+dtheta*float(i-1))
thesas(i)= theta/rad
p0=-x*sin(thetasas(i))+y*cos(thetasas(i))
dp=dn !*sin(thetasas(i))
if(theta.eq.90.0) then
p0=y
end if
k=nr*(i-1)
h(i)=1/(4.0*dp**2)
do j=2,nr-1,2
m = j-1
h(j)=1/(m**2*pi**2*dp**2)
h(j+1)=0.0
end do
shepp(i)=2.0/(pi*dp)**2
do j=1, nr-1, 1
shepp(j+1)= -2.0/(pi**2*dp**2*(2*j)**2-1.0))
end do
!do j=1,nr-1
! write(*,*) j,h(j)
!end do
!read(*,*) mm
do j=1,nr
Iptheta(j)=Iptheta(k+j)

r(j)=1.0*(float(j-1)*dr-Radius)

if(theta .eq. 90) then
  r(j)=1.0*(float(j-1)*dr-Radius)
end if
end do

call sorting2(nr,r,IPtheta)
do j=1,nr
  Ptheta(j,j)=Iptheta(j)
end do
do j=1,nr
do m=1,nr
  mm=abs(j-m)
  Qtheta(j,m)=Qtheta(j,m)+h(mm+1)*Ptheta(m,j)
end do
Qtheta(j,j)=dp=Qtheta(j,1)
end do
end do

g=0.0
do i=1,nt
  p0= y*cos(thetas(i))-x*sin(thetas(i))
  ! write(8,*), x,y,p0
  if(theta .eq. 90) then
    p0=y
  end if
  do j=1,nr
    Q(j)=Qtheta(j,i)
    ! write(8,*) Q(j)
    r(j)=1.0*(float(j-1)*dr-Radius)
  end do
  if(thetas(i) .eq. 90) then
    r(j)=1.0*(float(j-1)*dr-Radius)
  end if
end do
call sorting2(nr,r,Q)
call spline (r, Q, b, c, d, nr)
g=g+spline(p0, r, Q, b, e, d, nr)
end do
g=pi*g/nt

else
g=0.0
end if
gxy(iy)=g/(2.0*pi**2.0)
gps(iz,iy)=g/(2.0*pi**2.0))
Agps(ix,iy)=exp(-((x**2+y**2))
end do
write(9,200)(gxy(iy),iy=1,ny)
write(12,200)(Agps(ix,iy),iy=1,ny)

end do

!direct integration

call directKgxy, Radius, thetamin, dtheta, dr, nx, ny)
stop

200 format(161(e15.4))
201 format('p0=',f15.6,/,e15.4)

END PROGRAM Radontransform

double precision function fp(p)

use inputdata
use spllning
implicit none

integer i
double precision p
!double precision b(nr),c(nr),d(nr)

double precision ispline,deriv3,deriv4

call spline (r, Itheta, b, c, d, nr)
fp=deriv3(p, r, Itheta, nr, 1)
fp=ispline(p, r, Itheta, b, c, d, nr)
fp=deriv4(p, r, Itheta, nr, 3, 1)
fp=pfp/(p-p0) **2((p-6.0))
end function fp

double precision function ftheta(theta)

use inputdata
use splining

implicit none

double precision thetai !, b(nt), c(nt), d(nt)

double precision ispline

!call spline (thetas, Itheta, b, c, d, nt)
ftheta=ispline(thetai, thetas, Itheta, b, c, d, nt)
end function ftheta

end function ftheta

double precision function fs(s)

use direct
use splining

implicit none

integer i

double precision s

double precision ispline, deriv3, deriv4

!fs=deriv3(s, ss, Is, iss, 1)
fs=ispline(s, ss, Is, b, c, d, iss)
!fp=deriv4(s, ss, Is, iss, 3, 1)
end function fs

end function fs

subroutine reading(intheta)

! reading population data from a file and rewriting it to one array theta

use inputdata

implicit none

integer i, j, k, kk

double precision intheta(nr*nt), IO(nr, nt)

open(unit=7, file='smoothdata714.dat') !706inputdata
!open(unit=7, file='714inputdata.dat')
open(unit=8, file='testread.dat')

IO=0.0

do i=1, nr
    read (7, *) (IO(i, j), j=1, nt)
    write (8, 100) (IO(i, j), j=1, nt)
end do

do i=1, nt

do j=1, nr
    k=nr*(i-1)+j
    intheta(k)=IO(j, i)
end do
end subroutine reading

subroutine quanc8(fun,a,b,abserr,relerr,results,errest,nofun,flag)

! estimate the integral of fun(x) from a to b
! to a user provided tolerance.
! an automatic adaptive routine based on
! the 8-panel newton-cotes rule.
!
! input ..
!
! fun the name of the integrand function subprogram fun(x).
! a the lower limit of integration.
! b the upper limit of integration.(b may be less than a.)
! relerr a relative error tolerance. (should be non-negative)
! abserr an absolute error tolerance. (should be non-negative)
!
! output ..
!
! result an approximation to the integral hopefully satisfying the
! least stringent of the two error tolerances.
! errest an estimate of the magnitude of the actual error.
! nofun the number of function values used in calculation of result.
! flag a reliability indicator. if flag is zero, then result
! probably satisfies the error tolerance. if flag is
! xxx.yyy , then xxx = the number of intervals which have
! not converged and 0.yyy = the fraction of the interval
! left to do when the limit on nofun was approached.

double precision fun, a, b, abserr, relerr, results, errest, flag
integer nofun

double precision wO,w1,w2,w3,w4,area,z0,z0,z0,etcme,step,cor1l,templ
double precision qprev,qnov,qdiff,qleft,errest,relerr
double precision qright(31),f(16),x(16),fsave(8,30),xsave(8,30)
double precision dabs,dmaxl
integer levmin,levmax,levout,nomax,nofin,lev,nim,i,j
!
! *** stage 1 *** general initialization
! set constants.
!
levmin = 1
levmax = 30
levout = 6
nomax = 5000
nofin = nomax - 8*(levmax-levout+2**(levout+1))
!
! trouble when nofun reaches nofin
!
wO = 3956.040 / 14175.040
w1 = 23862.040 / 14175.040
w2 = -3712.040 / 14175.040
w3 = 41984.040 / 14175.040
w4 = -18160.040 / 14175.040
!
! initialize running sums to zero.
!
flag = 0.040
results = 0.040
**Corollary 1**

\[ \text{cor11} = 0.0d0 \]

**Error**

\[ \text{errest} = 0.0d0 \]

**Area**

\[ \text{area} = 0.0d0 \]

**Initializations**

\[ \text{nofun} = 0 \]

if (a .eq. b) return

\[
\text{*** stage 2 *** initialization for first interval}
\]

\[ \text{lev} = 0 \]

\[ \text{nim} = 1 \]

\[ x0 = a \]

\[ x(16) = b \]

\[ \text{qprev} = 0.0d0 \]

\[ f0 = \text{fun}(x0) \]

\[ \text{stone} = (b - a) / 16.0d0 \]

\[ x(8) = (x0 + x(16)) / 2.0d0 \]

\[ x(4) = (x0 + x(8)) / 2.0d0 \]

\[ x(12) = (x(8) + x(16)) / 2.0d0 \]

\[ x(2) = (x0 + x(4)) / 2.0d0 \]

\[ x(6) = (x(4) + x(8)) / 2.0d0 \]

\[ x(10) = (x(8) + x(12)) / 2.0d0 \]

\[ x(14) = (x(12) + x(16)) / 2.0d0 \]

do 25 j = 2, 16, 2

\[ f(j) = \text{fun}(x(j)) \]

25 continue


\[
\text{nfun} = 9
\]

\[
\text{*** stage 3 *** central calculation}
\]

\[ \text{requires qprev,x0,x2,x4,...,x16,f0,f2,f4,...,f16.} \]

\[ \text{calculates x1,x3,...,x15, f1,f3,...,f15,qleft,qright,qnow,qdiff,area.} \]

\[ x(1) = (x0 + x(2)) / 2.0d0 \]

\[ f(1) = \text{fun}(x(1)) \]

do 35 j = 3, 15, 2

\[ x(j) = (x(j-1) + x(j+1)) / 2.0d0 \]

\[ f(j) = \text{fun}(x(j)) \]

35 continue

\[ \text{nfun} = \text{nfun} + 8 \]

\[ \text{step} = (x(16) - x0) / 16.0d0 \]

\[ \text{qleft} = (w0*fun(f0)+w1*fun(f1)+w2*fun(f2)+w3*fun(f3)+w4*fun(f4))*\text{step} \]

\[ \text{qright} = (w0*fun(f(8))+w1*fun(f(9))+w2*fun(f(10))+w3*fun(f(11))+w4*fun(f(12)))*\text{step} \]

\[ \text{qnow} = \text{qleft} + \text{qright}(\text{lev}+1) \]

\[ \text{qdiff} = \text{qnow} - \text{qprev} \]

\[ \text{area} = \text{area} + \text{qdiff} \]

\[ \text{*** stage 4 *** interval convergence test} \]

\[ \text{esterr} = \text{dabs(qdiff)} / 1023.0d0 \]

\[ \text{tolerr} = \text{dmax1}(|\text{abserr}, \text{relerr} | \text{dabs(area)}) \times (\text{step/stone}) \]

if (lev .lt. levmin) go to 50

if (lev .ge. levmax) go to 62

if (nofun .gt. nof) go to 60

if (esterr .le. tolerr) go to 70

\[ \text{*** stage 5 *** no convergence} \]

\[ \text{locate next interval.} \]

50

\[ \text{nim} = 2*\text{nim} \]

\[ \text{lev} = \text{lev}+1 \]

store right hand elements for future use.

do 52 i = 1, 8
fsave(i,lev) = f(i+8)
xsave(i,lev) = x(i+8)
52 continue

! assemble left hand elements for immediate use.
!
qprev = qleft
do 55 i = 1, 8
  j = -i
  f(2*j+18) = f(j+9)
x(2*j+18) = x(j+9)
55 continue

! *** stage 6 *** trouble section
! number of function values is about to exceed limit.
!
60 nofin = 2*nofin
levmax = levout
flag = flag + (b - x0) / (b - a)
go to 70
!
! current level is levmax.
!
62 flag = flag + 1.0d0
!
! *** stage 7 *** interval converged
! add contributions into running sums.
!
70 results = results + qnov
  errest = errest + estarr
  corll = corll + qdiff / 1023.0d0
!
! locate next interval.
!
72 if (nim .eq. 2*(nim/2)) go to 75
  nim = nim/2
  lev = lev-1
go to 72

75 nim = nim + 1
if (lev .le. 0) go to 80
!
! assemble elements required for the next interval.
!
qprev = qright(lev)
x0 = x(16)
f0 = f(16)
do 78 i = 1, 8
  f(2*i) = fsave(i,lev)
x(2*i) = xsave(i,lev)
78 continue

! *** stage 8 *** finalize and return
!
80 results = results + corll
!
! make sure errest not less than roundoff level.
!
if (errest .eq. 0.0d0) return
82 temp = dabs(results) + errest
if (temp .ne. dabs(results)) return
  errest = 2.0d0*errest
go to 82
end

=====================================================================
subroutine spline (x, y, b, c, d, n)
Calculate the coefficients $b(i)$, $c(i)$, and $d(i)$, $i=1,2,...,n$
for cubic spline interpolation

$$s(x) = y(i) + b(i)(x-x(i)) + c(i)(x-x(i))^2 + d(i)(x-x(i))^3$$
for $x(i) \leq x \leq x(i+1)$

Alex G: January 2010

input...

x = the arrays of data abscissas (in strictly increasing order)
y = the arrays of data ordinates
n = size of the arrays $x(*)$ and $y(*)$ ($n \geq 2$)
output...
b, c, d = arrays of spline coefficients

comments ...
spline.f90 program is based on fortran version of program spline.f
the accompanying function fspline can be used for interpolation

--------------------------------------------------

! implicit none
integer n
double precision x(n), y(n), b(n), c(n), d(n)
integer i, j, gap
double precision h

gap = n-1
! check input
if ( n < 2 ) return
if ( n < 3 ) then
  b(1) = (y(2)-y(1))/(x(2)-x(1)) ! linear interpolation
  c(1) = 0.
  d(1) = 0.
  b(2) = b(1)
  c(2) = 0.
  d(2) = 0.
  return
end if

! step 1: preparation
! d(1) = x(2) - x(1)
c(2) = (y(2) - y(1))/d(1)
do i = 2, gap
  d(i) = x(i+1) - x(i)
b(i) = 2.0*d(i-1) + d(i)
c(i+1) = (y(i+1) - y(i))/d(i)
c(i) = c(i+1) - c(i)
end do
!
! step 2: end conditions
!
b(1) = -d(1)
b(n) = -d(n-1)
c(1) = 0.0
c(n) = 0.0
if(n /= 3) then
  c(1) = c(3)/(x(4)-x(2)) - c(2)/(x(3)-x(1))
c(n) = (c(n-1)/(x(n)-x(n-2)) - c(n-2)/(x(n-1)-x(n-3))
c(1) = c(1)*d(1)**2/(x(4)-x(1))
c(n) = c(n)*d(n-1)**2/(x(n)-x(n-3))
end if
!
! step 3: forward elimination
!
do i = 2, n
  h = d(i-1)/b(i-1)
b(i) = b(i) - h*d(i-1)
c(i) = c(i) - b*c(i-1)
end do

--------------------------------------------------
! step 4: back substitution
! c(n) = c(n)/b(n)
do j = gap, 1, n-1
   c(i) = (c(i) - d(i)*c(i+1))/b(i)
end do
!
! step 5: compute spline coefficients
!
b(n) = (y(n) - y(gap))/d(gap) + d(gap)*(c(gap) + 2.0*c(n))
do i = gap, 1, n-1
   b(i) = (y(i+1) - y(i))/d(i) - d(i)*(c(i+1) + 2.0*c(i))
   d(i) = (c(i+1) - c(i))/d(i)
   c(i) = 3.0*c(i)
end do
!
c(n) = 3.0*c(n)
d(n) = d(n-1)
end subroutine spline

function ispline(u, x, y, b, c, d, n)
function evaluates the cubic spline interpolation at point z
ispline = y(i)+b(i)*(u-x(i))+c(i)*(u-x(i))**2+d(i)*(u-x(i))**3
where x(i) < u < x(i+1)

! input:
! u  the abscissa at which the spline is to be evaluated
! x, y = the arrays of given data points
! b, c, d = arrays of spline coefficients computed by spline
! n  = the number of data points

! output:
! ispline = interpolated value at point u

implicit none
double precision ispline
integer n
double precision u, x(n), y(n), b(n), c(n), d(n)
integer i, j, k
do double precision dx
!
if u is outside the x() interval take a boundary value (left or right)
if(u <= x(1)) then
   ispline = y(1)
   return
end if
if(u >= x(n)) then
   ispline = y(n)
   return
end if
!
! binary search for for i, such that x(i) <= u <= x(i+1)
!
i = 1
j = n+1
do while (j > i+1)
   k = (i+j)/2
   if(u < x(k)) then
      j = k
   else
      i = k
   end if
end do
!
! evaluate spline interpolation
dx = u - x(i)
isepline = y(i) + dx*(b(i) + dx*(c(i) + dx*d(i)))
end function isepline

function deriv3(xx, xi, yi, ni, m)
! Evaluate first- or second-order derivatives
! using three-point Lagrange interpolation
! written by: Alex Godunov (October 2009)

implicit none
integer, parameter :: n=6
double precision deriv3, xx
integer ni, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
integer i, j, k, ix

! exit if too high-order derivative was needed,
if (m > 2) then
  deriv3 = 0.0
  return
end if

! if x is outside the xi(1)-xi(ni) interval set deriv3=0.0
if (xx < xi(1) .or. xx > xi(ni)) then
  deriv3 = 0.0
  return
end if

! a binary (bisectional) search to find i so that xi(i-1) < x < xi(i)
i = 1
j = ni
do while (j > i+1)
k = (i+j)/2
if (xx < xi(k)) then
  j = k
else
  i = k
end if
end do

! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...
i = i + 1 - n/2

! check boundaries: if i is outside of the range [1, ... n] -> shift i
if (i < 1) i=1
if (i + n > ni) i=ni-n+1

! old output to test i
! write(*,100) xx, i
! 100 format (f10.5, 15)
! just wanted to use index i
ix = i

! initialization of f(n) and x(n)
do i=1,n
  f(i) = yi(ix+i-1)
  x(i) = xi(ix+i-1)
end do

! calculate the first-order derivative using Lagrange interpolation
if (m == 1) then
  deriv3 = (2.0*xx - (x(2)+x(3))*f(1))/((x(1)-x(2))*(x(1)-x(3)))
  deriv3 = deriv3 + (2.0*xx - (x(1)+x(3))*f(2))/((x(2)-x(1))*(x(2)-x(3)))
  deriv3 = deriv3 + (2.0*xx - (x(1)+x(2))*f(3))/((x(3)-x(1))*(x(3)-x(2)))
else
  deriv3 = 2.0*f(1)/((x(1)-x(2))*(x(1)-x(3)))
  deriv3 = deriv3 + 2.0*f(2)/((x(2)-x(1))*(x(2)-x(3)))
  deriv3 = deriv3 + 2.0*f(3)/((x(3)-x(1))*(x(3)-x(2)))
end if
end function deriv3

function deriv4(xx, xi, yi, ni, n, m)
Evaluate first- or second-order derivatives
on three or four data points
using interpolation based on divided differences
written by: Alex Godunov (October 2009)

! input ...
i xi() - the abscissa at which the derivative is to be evaluated
i yi() - the arrays of data abscissas
i ni - size of the arrays xi() and yi()
! n - number of points to evaluate derivatives
i m - order of a derivative (1 or 2)
! output ...
i deriv4 - the first- or second-order derivative

implicit none
double precision deriv4, xx
integer ni, n, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
double precision d1, d2, d3, h, s
integer i, j, k, ix

! exit if too high-order derivative was needed,
! or too many base points were needed for derivatives
if (m > 2 .or. m > n .or. n > 5) then
  deriv4 = 0.0
  return
end if

! if x is outside the xi(i)-xi(ni) interval set deriv4=0.0
if (xx < xi(i) .or. xx > xi(ni)) then
  deriv4 = 0.0
  return
end if

! a binary (bisectional) search to find i so that xi(i-1) < x < xi(i)
i = 1
j = ni
do while (j > i+1)
k = (i+j)/2
if (xx < xi(k)) then
  j = k
else
  i = k
end if
end do

! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...
i = i + 1 - n/2

! check boundaries: if i is outside of the range [1, ... n] -> shift i
if (i < 1) i=1
if (i + n > ni) i=ni-n+1

! old output to test i
write(*,100) xx, i
! 100 format (f10.5, 15)
! just want to use index i
ix = i

! initialization of f(n) and x(n)
do i=1,n
  f(i) = yi(ix+i-1)
x(i) = xi(ix+i-1)
end do

! calculate divided difference coefficients
  d1 = f(2) - f(1)
if (n > 2) d2 = f(3) - 2.0*f(2) + f(1)
if (n > 3) d3 = f(4) - 3.0*f(3) + 3.0*f(2) - f(1)
h = x(2) - x(1)
s = (xx - x(1))/h

! calculate the first order derivative
if (m i) then
  deriv4 = (1.0/h)*d1
if (n > 2) deriv4 = deriv4 + (1.0/h)*(2.0*s-1.0)/2.0)*d2
if (n > 3) deriv4 = deriv4 + (1.0/h)**((3.0*s*s-6.0*s+2.0)/6.0)*d3
end if

! calculate the second order derivative
if (m w. and. n > 2) then
  deriv4 = (1.0/h**2)*d2
if (n > 3) deriv4 = deriv4 + (1.0/h**2)*(s-1.0)*d3
end if

end function deriv4

!**************************************************************************************
! Function gauss16(f,a,b)
!******************************************************************************
! Integration of f(x) on [a,b]
! Method: Gauss 16 points
! written by: Alex Godunov (October 2009)
!******************************************************************************
!
! IN:
f - Function to integrate (supplied by a user)
a - Lower limit of integration
b - Upper limit of integration
! OUT:
gauss16 - Result of integration
!******************************************************************************

implicit none
integer, parameter :: n=8
double precision gauss16, f, a, b
double precision ti(n), ci(n)
data ti/0.0950125098, 0.2816035507, 0.4580167776, 0.6178762444, &
SUBROUTINE Gaussian16(r,m,c)
   ! This subroutine is designed to compute the Gaussian integral
   ! using a simple recursive formula.
   ! Input:
   ! r - Start point of the integration interval
   ! m - Midpoint of the integration interval
   ! c - End point of the integration interval
   ! Output:
   ! r - The computed Gaussian integral

   r = 0.0;
   m = (b-a)/2.0;
   c = (b+a)/2.0;
   DO i = 1,n
      r = r + ci(i) * (f(m*(-1.0)*ti(i) + c) + f(m*ti(i) + c))
   END DO
   gauss16 = r*m
   return
end function gaussian16

SUBROUTINE Gaussian24(f,a,b,I)
   ! This subroutine is designed to compute the Gaussian integral
   ! using Gaussian Quadratures for 24 points.
   ! Input:
   ! f - Integrand provided by user
   ! a - Lower interval bound
   ! b - Upper interval bound
   ! I - Value of the given interval
   ! Output:
   ! r - The computed Gaussian integral

   IMPLICIT NONE
   DOUBLE PRECISION a, b, f, I, c, m
   DOUBLE PRECISION, dimension (12) :: Xi = (/0.0640568929, 0.1911188675, 0.3150426797, 0.4337935076, 0.5454214714, 0.6480936519, 0.7401241916, 0.8200019859, 0.8864155270, 0.9382745520, 0.9747285559, 0.9951872199/)
   DOUBLE PRECISION, dimension (12) :: Wi = (/0.1279381953, 0.1258374563, 0.1216704729, 0.1155056680, 0.1074442701, 0.0978180521, 0.0861901615, 0.0733464814, 0.0592968649, 0.0442774388, 0.0285313886, 0.0123412297/)
   INTEGER*4 j
   ! DO i = 1,12
   !    r = r + ci(i) * (f(m*(-1.0)*ti(i) + c) + f(m*ti(i) + c))
   ! END DO
   ! return
end subroutine Gaussian24

SUBROUTINE directKgf(Radius, thetamin, dtheta, dr, nx, ny)
   ! This subroutine is designed to compute the direct KGF integrals
   ! using recursive formulas.
   ! Input:
   ! Radius - Radius of the integration interval
   ! thetamin - Minimum angle of the integration interval
   ! dtheta - Angle increment of the integration interval
   ! dr - Radial increment of the integration interval
   ! nx - Number of angles
   ! ny - Number of radial points
   ! Output:
   ! r - The computed direct KGF integral

   ! DO i = 1,nx
   !    ! DO j = 1,ny
   ! END DO
   ! return
end subroutine directKgf
use inputdata
use splining
use direct

implicit none

!common

integer nx,ny

double precision, parameter:: rad = 57.2958
double precision, parameter :: pi = 3.141592653

integer n

!real Radius for qase

double precision Radius,thetamin, dtheta, dx, dy

common/parameters/dx,dy
integer nofun

double precision errest, flag

! working

integer i, j, it, ip, iis, ix, iy

double precision s(1000),IsA(1000), smax, smin, ds, Its

double precision x(1000), y(1000), xx(nx), yy(ny)

double precision g(x,y), f(nr,nt), fA(nr,nt) ! change f(101,179), fA(101,179)

double precision delta, di, dii, dn, p

external fs

open(unit=10, file='direct.dat')
open(unit=11, file='analytical.dat')

n=nx

do i = 1,nx,1

    xx(i) = dx*(i-1)-Radius
    yy(i) = dy*(i-1)-Radius

end do

dn=2*Radius/(n-1)

do iis = 1,n,1

    x(iis) = dn*(iis-1)-Radius

end do

do it=1,nt,1 !change from to 179,1

    theta= thetamin+dtheta*(it-1) !it

do ip=1,nr,1 ! change from nr to 101

write(*,*),it,ip

    p=-((dx*(ip-1)-Radius)
do iis = 1, n, 1
   y(iis) = x(iis)*tan(theta/rad) - p/cos(theta/rad)
   if(theta.eq.90) then
      y(iis) = p
   end if
end do
if(theta.eq.90 .and. abs(p).ge.1.6) then
   smax = 0.0001
else
   smax = (Radius**2-p**2)**0.5
end if
smin = smax
write(*, smin, smax

ds = (2.0*Radius)/n
s = 0.0
write(10,100) theta, p, smin, smax

do iis = 1, n, 1
   s(iis) = x(iis)*cos(theta/rad) + y(iis)*sin(theta/rad)
   if(theta.eq.90) then
      s(iis) = x(iis)
   end if
   write(*, *) iis, s(iis), smin, smax
   if(s(iis).ge.smin .and. s(iis).le.smax) then
      call binary(y(iis), yy, ny, i)
      write(*, *) s(iis), y(iis), yy(i), i
      di = abs(y(iis)-yy(i)) ! left
      dii = abs(y(iis)-yy(i+1)) ! right
      if(di.gt.dii) then
         j = i+1  ! closer to the right
         delta = dii
         write(*, *) 'di+', delta, j
      else
         j = i  ! closer to the left
         delta = di
      end if
   end if
end do
!write(*,*) 'di', delta, j
end if
if(abs(delta/yy(j)) .lt. 1.E-1) then
  ix=iis
  iy=j
  iss=iss+1
  Is(iss)=gf(ix,iy) !Is(iss)=IsA(iis)
  ss(iss)=s(iis) ! ss(iss)=s(iis)
  if(theta.eq.90) then
    Is(iss)=gf(iy,ix) !Is(iss)=IsA(iis)
    ss(iss)=s(iis) ! ss(iss)=s(iis)
  end if
end if
end if
end do
! call sorting2(iss,ss,Is)
do i=1,iss
  !write(10,102) i, ss(i), Is(i)
!end do
if(iss.gt.2) then
  call spline (ss, Is, b, c, d, iss)
call quanc8(fs,smin,smax,abserr,relerr,Its,errest,nofun,flag)
else
  Its=0.0
end if
f(ip, it)=Its
fA(ip, it)=(pi**0.5)*exp(-p**2)
write(*,*) 'quanc', ip, it, f(ip, it), fA(ip, it)
end do
do ip=1,nr,1 ! change from nr to 100
write(10,1000)(f(ip, it), it=1, nt) ! change from it=1, nt to it=1,179
write(11,1000)(fA(ip, it), it=1, nt)
end do

100 format(161(e15.4)) ! change from 161 to 179
101 format (9x,'theta',10x, 'p',15x, 's',14x, 'x',14x, 'y', 5x,'ix',3x, 'iy',2x, 'iss',5x, 'ss(iss)',8x, 'Is(iss)', 7x,'gf(ix,iy)')
end subroutine directI
!======================================================================

subroutine binary(xx,xi,ni,i)
!======================================================================

implicit none

integer ni,i,j,k

double precision xx,xi(ni)

! a binary (bisectional) search to find i so that xi(i) < x < xi(i+1)
! i = 1
! j = ni
! do while (j > i+1)
! k = (i+j)/2
! if (xx < xi(k)) then
! j = k
! else
! i = k
! end if
! end do
end subroutine

!======================================================================

subroutine sorting2(n,xd,yd)
!======================================================================

! Sorts the arrays in descending order and shuffles the corresponding
data at the same time
! written by: Ana Samolov
! same as sorting just customise for specific data types
!======================================================================

implicit none

integer i,n, swap

double precision xd(n),yd(n)
double precision tempx,tempy

swap=1

do while (swap.eq.1)

swap=0

do i=1,n-1

if(xd(i).gt.xd(i+1)) then

  tempx=xd(i+1)
  xd(i+1)=xd(i)
  xd(i)=tempx

  tempy=yd(i+1)
  yd(i+1)=yd(i)
  yd(i)=tempy

  swap=1

end if
Nd
end do
end do
end subroutine sorting2
VITA

Ana Samolov
Department of Physics
Old Dominion University
Norfolk, VA 23529

EDUCATION

May 2012  Ph.D., Physics, Old Dominion University
Dissertation: *Case Studies in Many-Body Physics*

May 2008  M.S., Physics, Old Dominion University

Dec 2005  B.S., Applied Physics and Informational Science, University of Belgrade
Senior Thesis: *The Simulation of Franck-Hertz experiment*

HONORS AND AWARDS

Nov 2011  64th Annual Gaseous Electronics Conference, Student Travel Award

Aug 2011  Old Dominion University Fellowship for 2011/2012 school year

Mar 2011  Particle Accelerator Conference Travel Award

Jan 2011  US Particle Accelerator School Financial Support

Oct 2010  Old Dominion University Graduate Student Travel Award

Oct 2010  National Science Foundation Student Travel Award

Oct 2009  62nd Annual Gaseous Electronics Conference, Student Travel Award

Aug 2007  The Department of Physics Award for Most Distinguished Scores on the Qualifying Exams in Physics


Typeset using \LaTeX.