High Performance Computation and Database of Radiative Properties with an Interface for ICF Applications

Jiankui Yuan

December 2001

UWFDM-1164

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

HIGH PERFORMANCE COMPUTATION AND DATABASE OF RADIATIVE PROPERTIES WITH AN INTERFACE FOR ICF APPLICATIONS

by

JIANKUI YUAN

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN – MADISON

2001

Chapter 1

Introduction

This thesis introduces the creation and use of an object-oriented relational database containing radiative properties of hot dense plasmas in radiation hydrodynamics calculations to simulate laser fusion experiments and in spectroscopic analysis of laser fusion experiments. Atomic physics computations are done in a distributed environment. Technologies such as MPI, CORBA and EJB are involved. The Oracle 8i object oriented relational database management system is used to serve the data and Java technology is used to construct a portal for the plasma physics user to access and use the data. This supports dense plasma physics research in the Fusion Technology Institute (FTI) at the University of Wisconsin and other organizations if they so desire.

The layout of the thesis is as follows: In Chapter 1, we start with the discussion of difficulties in large-scale atomic data calculations and the goals of the thesis. In Section 1.2, we give a brief overview of various atomic models in hot dense plasmas. In Section 1.3, we introduce high performance computing technologies using parallel computing infrastructure such as MPI, OpenMP, and PVM, which are usually adopted in

scientific computing environment, and distributed object computing such as CORBA, EJB, which are commonly used in the commercial systems.

In Chapter 2, we present more details of the information technologies that are used in the thesis. Starting with the comparison of parallel and distributed computing, the description of the architectures and main components of the two most popular distributed object-oriented computing frameworks CORBA and EJB are then provided. Several aspects of Java technology such as thread, networking and JDBC are also touched. Finally, the Oracle DBMS, which serves as the database manager, is discussed.

Chapter 3 covers the physics underlying the various atomic models used in the thesis. First, the calculation of atomic structure in the non-relativistic theory is discussed. From the Hamiltonian operator to the antisymmetrization of the electron waves and to the Slater integrals and the energy matrix, we describe the conventional ways of treating atomic structure under perturbation theory. Then, many atomic processes are discussed and some formulas used in the program are provided. In Section 3.2, we give a description of the plasma models and various rate coefficients that are used in the current opacity NLTE model. In the following section, the RSSOPA model based on the UTA method is discussed.

In Chapter 4, we discuss the implementation of the atomic data distributed computing system. First, we verify the accuracy of ATBASE codes and the RSSUTA model. Then we demonstrate the importance of JJ coupling by studying the transition spectra patterns and comparing a Nb spectrum with experiment. The speed up of our

parallel implementation of the RSSUTA code is also presented. Finally, we describe the database and its graphic user interface.

In Chapter 5, we come to the conclusions. A User's Manual is also given in the Appendix.

1.1 Atomic Database Using Distributed Object Oriented Technology

A suite of codes named ATBASE [PW96] had been developed by Dr. P. Wang, Dr. J. J. MacFarlane, Dr. G. A. Moses and other members in the FTI. These codes have been successfully used to analyze the experiments of light ion fusion [CCW98], laser-driven inertial confinement fusion [MCW98] and other dense plasma research. The ATBASE codes consist of three parts: the atomic data generator, the plasma population modeling and the muffin-tin EOS calculation. The atomic data generator is based on Cowan's non-relativistic code [COW81]. It provides very accurate atomic data such as energy levels, transition oscillator strengths, photoexcitation cross sections and other radiative and collisional properties. The second part of ATBASE calculates the plasma state populations with several options: the LTE Saha model, the Coronal Equilibrium model and the full CRE steady state model. The EOS muffin-tin model includes sophisticated

treatments for both bound and free electron density distributions in the plasma by solving the self-consistent-field Dirac equation. It provides very good EOS data.

However, there are several areas that can be improved. First, ATBASE lacks a graphic interface to effectively assist users' access to the data. In ATBASE calculations, there are many important input parameters that users should be aware of and various computing models that are based on different physical assumptions. A graphic user interface helps users specify the problem, interact with the program dynamically and also help users analyze the output.

Secondly, the atomic data produced by ATBASE are stored in many scattered flat-files. Because of the drawbacks of using flat-files, such as management difficulties and less efficient IO operations, it is desirable to store these atomic data in a database management system (DBMS). In this way, we can avoid the time-consuming repeated computations for the same atomic data and the data diagnosis is also much easier. Once the database has been built up, various technologies such as CORBA, EJB and JSP can be applied to make the database accessible from the Web.

Thirdly, although ATBASE is able to provide accurate radiative properties such as transition energies and opacities using the UTA method, it is less accurate to apply to the high-Z spectroscopy analysis (see Fig. 4.1.3). The difficulty for ATBASE to simulate the high-Z spectra is the inherent complexity of enormous numbers of transition lines for high Z elements using the DTA method and the non-relativistic treatment of the UTA method. We have built a new model named RSSOPA (the

relativistic **s**ingle-configuration-**s**ingle-electron **opa**city model), which overcomes the complexity of enormous transition lines by using the UTA method but in the fully relativistic treatment, and therefore giving us more accurate results (see Fig. 4.1.3 and Fig. 4.1.4). In the RSSOPA model, the atomic data are in the JJ coupling approximation. Furthermore, the RSSOPA model is implemented for parallel computation using MPI.

The goals of the thesis are:

I. To build a distributed computing system for the high performance computing of atomic data.
II. To establish an atomic database to store the data such as EOS and opacities used in ICF applications.
III. To build a graphic user interface for ICF applications.
IV. To validate the atomic data in the database and perform spectrum analysis for medium to high Z elements using the RSSUTA model.

1.2 Atomic Physics in Hot Dense Plasmas

Atomic physics plays an important role in the investigation of the properties and behavior of hot laboratory and astrophysical plasmas containing highly stripped atoms. An understanding of the physical processes of the ionized atoms in hot dense plasmas has various applications in astrophysics, fusion research, X-ray laser and other branches of modern physics. Recent technical advances have made it possible to study the

spectroscopy from negative ions to hydrogen-like Uranium ion U^{+91}, as well as the collisional characteristics of the resonant processes arising from collisional excitation and dielectronic recombination [GPR97][PDS91].

The interests of atomic physics in hot plasmas are mainly in three general fields. The first is to study the influence of the plasma environment on the atomic structure, such as the bound electron wave function, energy levels and ionization degrees. The second is the study of the collision processes among electrons, photons and ions inside the plasma. These processes determine the charge and excited state distributions. The quantities to describe the probabilities of these processes are their cross-sections and rates. The third is the subject of the emission and absorption spectra of the plasma.

With the increase of plasma density, the atomic properties transition from the free atom model, which means the interaction among atoms is so weak that one atom moves as if no other atoms exist, to the screened electron cloud regime and then to the quasi-molecular regime. There are several models to address the effect of the plasma environment on the atomic structure.

- The Debye-Hückel (DH) model

The DH model is based on the Poisson equation and the Boltzmann statistical distribution for the ions and electrons. The important quantities in the DH model are the Debye screen length, Debye sphere and Debye screened potential. The Debye sphere defines the extent range the other ions can affect the central ion. The DH model is not valid at very high density.

- The Thomas-Fermi (TF) model

The TF model is based on the micro-view of the Poisson equation and the Fermi-Dirac (FD) statistics instead of the Boltzmann distribution. In this model, both bound and free electrons conform to the FD distribution and all electrons move in a confined sphere. The Thomas-Fermi-Dirac (TFD) model includes the corrections due to the exchange interaction of the electrons.

- The Ion Sphere (IS) model [RZ72][YSZ96]

This model is similar to the TF model except that it treats the bound electrons differently. For the bound electrons, the IS model uses the quantum mechanic Schrödinger equation instead of the statistical distribution. This model is also referred to as the average atom model, which has many applications in the EOS calculations and the analysis of complicated spectra of plasmas. This model can also treat the free electrons quantum mechanically by solving the wave equation.

- The Ion Correlation (IC) model [DP82][EKK76]

The IC model is the most adequate model to describe the atomic structure in plasmas. It includes the consideration of ion-ion correlation. This model needs to solve a hypernetted chain (HNC) equation to count the interaction of the ions besides the electron-ion interaction for the two-component plasma.

The effect of the environmental ions modifies the potential of the central ion with an additional repulsive potential and thereby reduces the binding energies of the electrons of the central ion. The two important phenomena of these effects are ionization

potential lowering or continuum lowering and atomic level shifting. Because of the complexity of involving the plasma effect into the atomic structure, people usually use the atomic data generated under the free atom assumption, which is the topic of Section 3.1.1.

There are a dozen important processes involved in the plasma interactions as listed in Table 3.2.1. The fully quantum mechanical treatments of these processes are so difficult that the empirical formulas or fitting formulas are often used, for example, the Lotz's formula [LZ68] for electron impact ionization, the Burgess-Merts' formula [BC83] for the dielectronic recombination and the Gaunt modification to the classical formulas. We discuss the various radiative, collisional and resonant processes in Section 3.1.2.

The radiation emitted from hot plasmas is an important diagnostics tool. The emitted radiation intensity and spectral distribution are determined by the plasma density and temperature. Through the analysis of the satellite line intensity ratios of the emission spectrum, we can infer the plasma condition [MMS94]. The typical emission spectra of highly ionized plasmas are shown in Fig. 1.1.1. The line-by-line modeling method [COW81] which is adequate for low-Z plasmas consisting of ions with a small number of bound electrons is not practical to interpret the cluster spectral structure coming from the high-Z highly stripped plasmas. The more powerful method is the UTA model, which is also a practical approximation to calculate the plasma opacity. We discuss this model in Section 3.2.

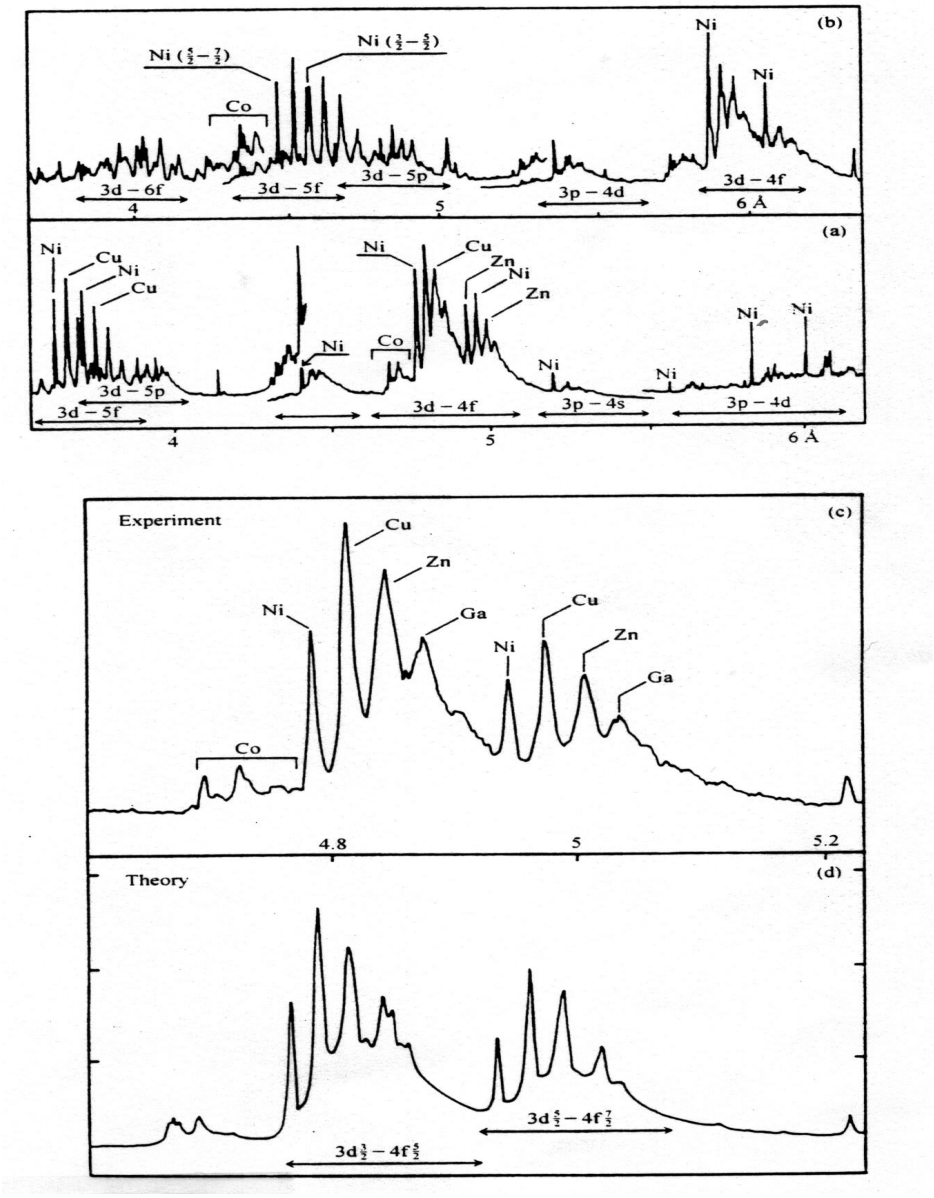


Fig. 1.1.1 Experimental gold (a) and tantalum (b) spectra from laser-produced plasmas. Comparison of the 4.7-5.2 Å region of the spectrum from the gold plasma (c) with computational results (d) yield the best fit for a 240eV ionization temperature [GMA86].

1.3 High Performance Computing: Parallel and Distributed

The speed of high performance computers continues to dramatically increase as a result of a hierarchy of processor parallelism and a hierarchy of memory access. On-chip parallelism in the circuitry of the processor functional units leads to superscalar performance by fetching and executing multiple instructions (~4) per clock cycle. Multi-level caches (L1 and L2 and sometimes L3) provide data locality to the processor registers that reduces fetches to main memory and improves processor performance. Shared memory parallelism (SMP) has several processors (~4-32) sharing data through the same main memory through high speed memory access technology. An alternative to this, massively parallel computers, have each processor with its own main memory and the processors share data through a network architecture where the program must explicitly request that data be retrieved from and sent to a distant processor. In the last five years, a hybrid high performance computing architecture that has clusters of SMP's connected by high speed interconnects, has gained favor. These can either be packaged together by a vendor such as the IBM SP computer or they can be"home built" from commodity chips such as the Intel Pentium and run the Linux operating system.

From the programmer's point of view, there are several levels of complexity involved in programming these computers. Using standard MPI [GLS99] or Java, distributed programming models run on SMP's. However, an alternative fine grained

programming model for SMP's called OpenMP allows the programmer to take advantage of the shared memory features. Mixed OpenMP and MPI programming models are just now being studied for their potential in improved performance.

The tools and standards for assembling distributed computing applications have been developed over the past 10 years. From the low-level data transmission APIs and protocols such as Remote Procedure Call (RPC) and Distributed Computing Environment (DCE) to object-based distributed schemes such as Common Object Request Broker Architecture (CORBA) [WOMG] and Remote Method Invocation (RMI) and OpenDoc, the distributed computing environment has evolved to a robust, platform-independent, flexible and extensible framework.

Java as one of the most important object-oriented (OO) programming language supports distributed computing. From low-level network communications to distributed objects and agents, Java offers an environment that encompasses various levels of distributed computing development. Java's API for socket URLs and other networking facilities is much simpler than what is offered by other programming languages like C and C++. Java RMI [WSUN] is a framework that allows objects to invoke methods on remote objects in the same way as methods of local objects. Java also has built-in support for writing multithread programs such as the synchronized keyword which is used to lock objects and classes to control concurrent access to data. The recent interests in using Java for scientific computing has even led to efforts to produce a message

passing interface to support parallel computation, such as mpiJava, an object-oriented Java interface to MPI.

The most important architectures that play major roles in the distributed object-oriented computing world are OMG's CORBA, Java RMI, EJB, Java IDL [WSUN] and Microsoft's DCOM. CORBA is a specification for a technology that allows objects on one machine to communicate with objects running on any number of different machines using different programming languages. The unique feature of Java RMI is its support for the dynamic code loading, the ability to download the bytecodes of an object's class if the class is not defined in the receiver's virtual machine. EJB, a framework for distributed object computing and specific to Java, becomes a powerful standard to build a web database-based distributed object computing system. A main task of this thesis is to create a system using CORBA and EJB against the Oracle DBMS. We further discuss these technologies in Chapter 2.

Chapter 2

Information Technology

In Chapter 2, we describe most of the information technologies that are used in the thesis. We begin with the comparison of distributed computing and distributed object computing and their architectures. In the following three sections, we first give a brief description of each technology we use in the thesis and then discuss how it is applied in the project. Some code fragments and results are also provided.

2.1 Overview of Distributed Computing and Distributed Object Computing

Parallel or distributed computing has become a key component of high performance computing. A parallel or distributed architecture is one that consists of a collection of processing units that cooperate to solve different parts of the problem simultaneously. According to the way that the computers communicate, the computer architectures are basically of two types: tightly coupled systems and loosely coupled systems [SN97]. Tightly coupled systems have a single system wide primary memory that is shared by all

the processors as shown in Fig. 2.1.1(a). Usually, tightly coupled systems are referred to as parallel processing systems. In loosely coupled systems, the processors do not share memory, and each processor has its own memory (Fig. 2.1.1(b)). So all physical communications between the processors are done by passing messages across the network that connects the processors. The loosely coupled systems are often referred to as distributed computing systems.

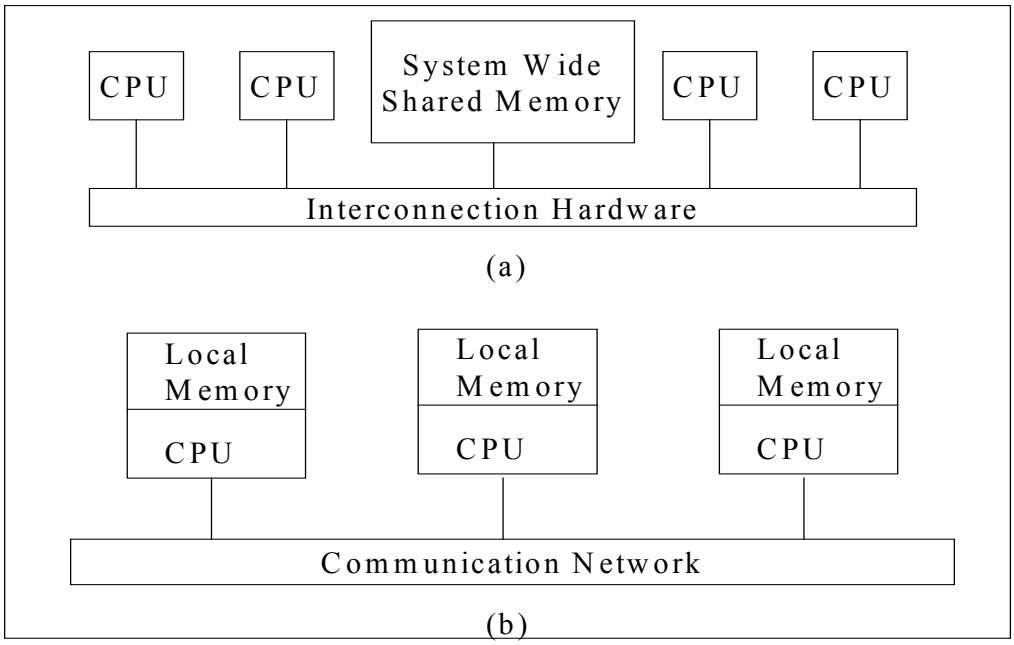


Fig. 2.1.1 Difference between tightly and loosely coupled multiprocessor systems: (a) tighly coupled system; (b) loosely coupled system.

Currently the most popular software environment that is used in the distributed computing is DCE (Distributed Computing Environment), which is defined by the Open Software Foundation (OSF). DCE is an integrated set of services and tools that can be

installed as a coherent environment on top of existing operating systems and serves as a platform for building and running distributed applications. The main components of DCE are threads package, remote procedure call (RPC) facility, distributed time service, name service and distributed file service. As shown in Fig. 2.1.2, the DCE software layer on top of the operating system and networking layer hides the differences between machines by automatically performing data-type conversions. Therefore, the heterogeneous nature of the system is transparent to the application programmers, making the distributed application development much simpler.

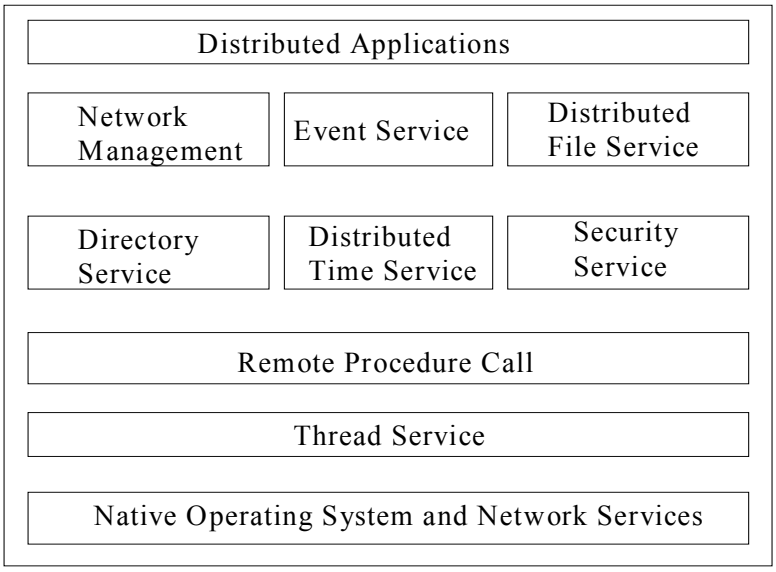


Fig. 2.1.2 OSF DCE Architecture

Another competing computing environment is the Object Management Group (OMG)'s Common Object Request Broker Architecture (CORBA). The most important difference between OSF DCE and OMG CORBA is their programming paradigms: DCE

was designed to support distributed procedural programming, while CORBA was designed to support distributed object-oriented programming. However, there are many similarities between DCE and CORBA. They both define an Interface Definition Language (IDL), and use IDL to define the interface that a server implements and is compiled into a client stub and a server skeleton. A client application calls the client stub to request a service, and then the client stub interfaces to the runtime system, which eventually invokes the server code that implements the requested service through the appropriate server skeleton. A high level architecture of CORBA is shown in Fig. 2.1.3.

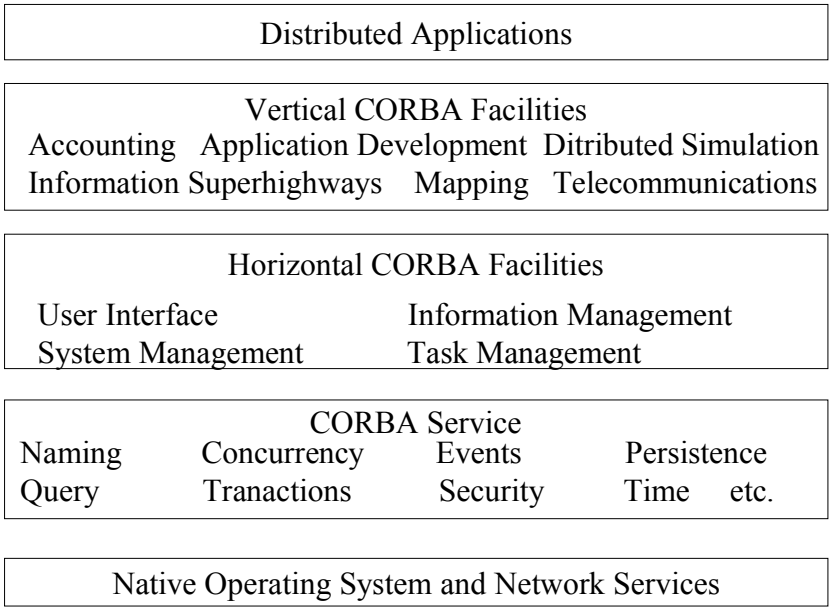


Fig.2.1.3 OMG CORBA Architecture

As shown in the above figures of DCE and CORBA architectures, both of them act as middleware layered between the applications layer and the operating system and networking layer. Their basic purposes are handling the transmission of service requests

and responses between clients and servers, shielding the user applications from concerns like the location of clients and servers on the network, differences between hardware platforms, operating systems, and implementation languages and networking protocols.

While DCE supports procedural programming, CORBA was designed to support Object-oriented (OO) programming. The characteristics of OO environment are: encapsulation of data and the functions that manipulate the data which enforces data integration and hiding; inheritance of interfaces and implementations which embodies the reusability of OO programming; polymorphism, which is the ability for a request for a specific operation to be handled differently depending on the type of object on which it is invoked. As one of several main OO programming languages, Java has built-in support for distributed computing. Java's API for sockets, URLs and other networking facilities is much simpler than that offered by other programming languages like C and C++.

The most common computing technology used in the scientific computing is MPI, which is the standard for distributed computing under DCE architecture. In this project, we use MPI to calculate all relativistic JJ coupling atomic data of the RSSOPA model. The computing flow chart is given in Section 3.2.2. For handling the graphic user interfaces and data management, the better choice is to use CORBA architecture, which supports object-oriented programming such as C++ and Java. This project uses CORBA technology to implement most of the computing. The Java programming language is

used. In the next three sections, we describe these technologies as they are in each project in the thesis.

2.2 Distributed Atomic Data Computing System Using CORBA and EJB

The distributed atomic data computing system is mainly CORBA-based. Because the CORBA components are portable across languages, operating systems and networks, it is a good choice for multi-tier database applications. There are two important projects, JEOSOPA and SPECTRA (see the list of projects) that use CORBA. In the following, we give a brief description of CORBA technology, and its application in these two projects. Description of EJB is also given, which is used in the project MIXOPA.

2.2.1 CORBA Architecture

Common Object Request Broker Architecture (CORBA) is a specification for a technology that allows objects on one machine to communicate with objects running on any number of different machines. It was designed to be the middleware glue allowing different languages to implement objects on different platforms. The Object bus provides an Object Request Broker (ORB) that lets clients invoke methods on remote objects either statically or dynamically. Fig. 2.2.1 shows the CORBA communication model.

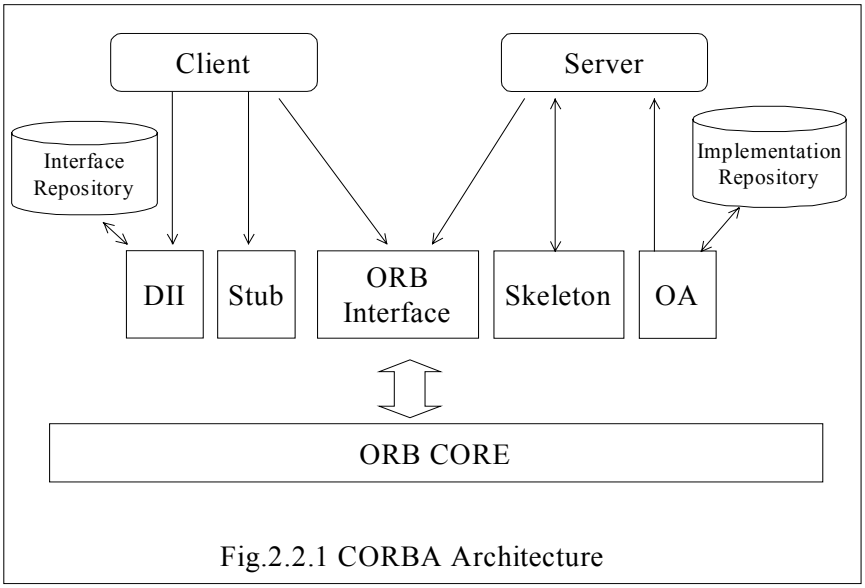


Fig.2.2.1 CORBA Architecture

The idea behind CORBA is to shield the developer from any of the low-level complexities of having one program communicate via a network to a program on another physical host. Using an ORB, a client object can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call and finds an object that can implement the request, passes it the parameters, invokes its method, and returns the results. The client does not have to be aware of where the object is located, its programming language or its operating system. It is the responsibility of the ORB to establish the remote communication with distributed objects and handle all network interactions in passing data between objects. The ORB itself is simply the software bus that can move messages between objects that are written in different languages on different hardware platforms implemented by different vendors.

The boundaries and the contractual interfaces between client and server are defined through IDL. IDL is a descriptive language that describes the interfaces being implemented by the remote objects. IDL defines the name of the interface, the names of each of the attributes and methods, the arguments for each method, and the return type. The specified IDL compiler maps the language neutral IDL into the native programming languages like C, C++ or Java. The separation of interface from implementation lets different languages communicate easily via CORBA.

The way that the IDL bridges the gap between client implementation and server implementation is through the generation of static stubs and skeletons. A stub is a client-side source file that implements a local proxy object, which defines how clients invoke corresponding services on the servers. The client interacts directly with the client stub. It is the responsibility of the client stub to make the invocation to the actual server object implementation. A skeleton refers to the server-side source that the server object implementation registers with. The skeleton's responsibility is to receive requests and dispatch these requests to the server object implementations. The process of creating client stub and server skeleton from an IDL file is shown below:

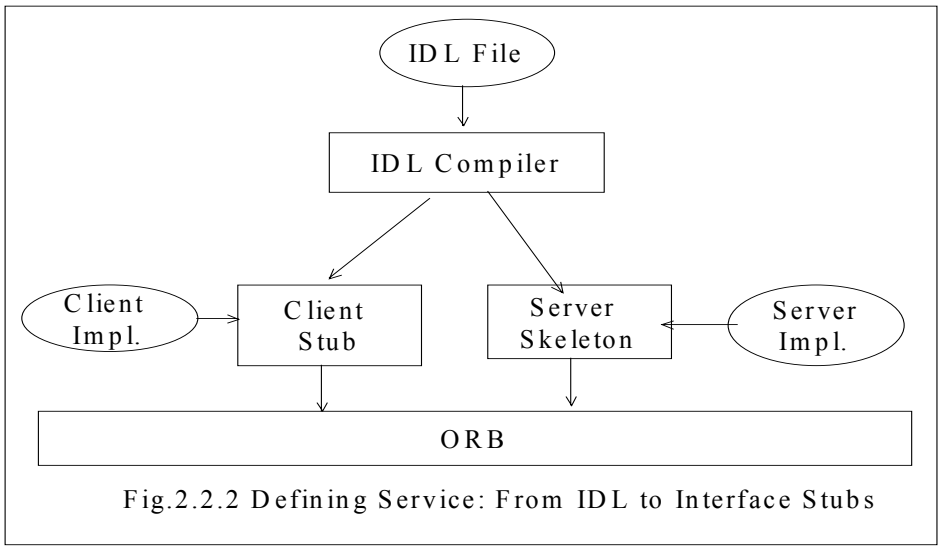


Fig.2.2.2 Defining Service: From IDL to Interface Stubs

Static stubs and skeletons must be generated at build time and compiled in with the source code. With the DII and DSI, it is not necessary to use IDL to generate static stubs and skeletons. The DII allows clients to query the ORB's Interface Repository (IR) for available objects and construct method request on the fly. The IR is a standard CORBA component, which contains meta-data describing the object available. From the IR, the client can determine what interfaces are available, and what their method, parameters and return types are, then it can dynamically create a method request using the DII. The server object receives the request having no knowledge of whether the client was built with static stubs or with the DII interface. Similarly, the server object does not need to be compiled in with a static skeleton to receive requests. The DSI automatically enables new objects to receive requests without having inherited from the IDL generated skeleton.

The Object Adapter (OA) is the main way by which object implementations access services via the ORB. It sits on top of the ORB's core communication services and accepts requests for service on behalf of the server's objects. When an object is created, the OA is responsible for creating a unique Internet Object Reference (IOR) and keeping a table of all registrations. Other objects use an object's IOR to locate and establish communication with it. The object implementation can also register policies under which the OA activates new instances and deactivates existing instances. The OA contacts with the IR that contains objects with all the information needed for the OA to activate object implementations. The OA keeps an internal reference count of activated instances. When the reference count has been reduced to zero, the instance is destroyed. CORBA specifies that each ORB must support a standard adapter called the Basic Object Adapter (BOA).

CORBA uses Internet Inter-ORB Protocol (IIOP), which specifies how General Inter-ORB Protocol (GIOP) messages are exchanged over a TCP/IP network. GIOP specifies a set of message formats and common data representations for communications between ORBs. The IIOP makes it possible to use the Internet itself as backbone ORB. It provides interoperations for TCP/IP based ORBs.

Projects implemented using CORBA

Project JEOSOPA performs the task of computing the EOS and opacity based on the code UTAOPA. Since UTAOPA is written in FORTRAN, we need to wrap it in a Java module using Java native interfaces. Java native interfaces allow a Java program to

call FORTRAN subroutines and pass through the argument parameters. JDBC is used to access the Oracle database for the atomic data, which is required in the opacity calculations.

The basic outline of JEOSOPA computing takes the following forms:

1. User sends the request that includes all needed information to perform the UTAOPA calculation.
2. The server object retrieves the needed atomic data from Oracle database through JDBC.
3. The Java version of UTAOPA is executed, and results are sent back to the user or saved into database if the user has permission.
4. User uses OPAVIEWER to view the output data.

Project SPECTRA performs the task of analyzing spectra from high-Z plasmas. Currently, only an LTE option is implemented. Because it involves intensive calculations, the module for spectrum computing is written in FORTRAN. Java native interfaces are also needed to cooperate with a Java program.

The basic outline of SPECTRA computing is as follows:

1. User specifies the plasma condition (single temperature and density pair is allowed) and invokes the spectrum calculation on the remote server.
2. The server opens a connection to the database, collects all needed atomic data and does LTE calculations.

3. User displays the spectrum result using Java 2D technology and analyzes the spectrum with the aid of atomic database.

4. If the user is not satisfied with the result, the user repeats the calculation by setting a different plasma condition.

2.2.2 Enterprise JavaBeans

Java has been recognized as an excellent platform for developing distributed server-side applications, producing implementation-independent abstractions for common enterprise technologies. For example, JDBC provides a vendor-independent Java interface for accessing SQL relational databases, JNDI (Java Naming and Directory Interface) provides an interface for abstracting directory services, EJB (Enterprise JavaBeans) [RM99] provides an abstraction for component transaction monitors (CTMs). Component transaction monitors provide a robust, component-based environment that simplifies distributed development while automatically managing the most complex aspects of enterprise computing, such as object brokering, transaction management, security, persistence, and concurrency.

EJB server-side components have two fundamentally different types: entity beans and session beans. Entity beans model real-world objects; these objects are usually persistent records in a database. Session beans are an extension of the client application and are responsible for managing processes or tasks. The activity that a session bean represents is fundamentally transient. It doesn't represent something in a

database. To implement an enterprise bean, two interfaces and one or two classes need to be defined:

(1) Remote interface. The remote interface for an enterprise bean defines the bean's business methods: the methods a bean presents to the outside world to do its work. The remote interface extends javax.ejb.EJBObject, which in turn extends java.rmi.Remote.

(2) Home interface. The home interface defines the bean's life cycle methods: methods for creating new beans, removing beans, and finding beans. The home interface extends javax.ejb.EJBHome, which also in turn extends java.rmi.Remote.

(3) Bean class. The bean class actually implements the bean's business methods. The bean class usually doesn't implement the bean's home or remote interfaces, but it must have methods matching the signatures of the methods defined in the remote interface and must have methods corresponding to some of the methods in the home interface. An entity bean must implement javax.ejb.EntityBean; a session bean must implement javax.ejb.SessionBean.

(4) Primary key. The primary key is a very simple class that provides a pointer into the database. Only entity beans need a primary key; the only requirement for this class is that it implements java.io.Serializable.

(5) Deployment descriptor. Deployment descriptors serve a function very similar to property files. They are used to customize behavior of enterprise beans at runtime without having to change the software itself. Once the deployment descriptor is complete and saved to a file, the bean can be packaged in a JAR file for deployment.

The JAR's path is given to the container's deployment tools, which read the JAR file. The container uses the deployment descriptor to learn about the beans contained in the JAR file and how to manage the bean at runtime.

EJB explicitly supports two mechanisms to manage large numbers of beans at runtime: instance pooling and activation. In the instance pooling, the EJB container creates several instances of a bean class and holds on in a pool. As clients make method requests, beans instance from the pool are assigned to the EJB object associated with the clients. When the EJB object doesn't need the instance any more, the instance returns to the instance pool. Instance pooling reduces the number of component instances and therefore resources needed to service client requests. It is less expensive to reuse pooled instances than to frequently create and destroy instances. For stateful session beans, they don't participate in instance pooling like stateless session beans and entity beans. Instead activation is used with stateful session beans to conserve resources. The EJB server can evict stateful session beans from memory by disassociating the stateful bean instance from its EJB object and serializing the bean's state to a secondary storage. Activating a bean is the act of restoring a stateful bean instance's state. When a method on the passivated EJB object is invoked, the container automatically instantiates a new instance and sets its field equal to the data stored during passivation.

Projects implemented using EJB

Project MIXOPA calculates the mixed opacities for arbitrary element combinations. All data for pure elements are stored in the database, and the mixed

opacities are calculated from these data for pure elements in a form of linear combination. This project is written in the pure Java programming language. The processing scenario of this project is:

1. EOS and opacity data are calculated using EOSOPA. The opacity grids for photon energy are 100 groups. These data are stored in the database.
2. User invokes the mixing method on the server after giving the mixing specification.
3. The server obtains the EOS and opacity data for each pure element, does the mixing calculation, and sends the results back to the user.
4. User saves the results on the local machine.

2.3 Java Thread, Network Programming and JDBC

This section gives descriptions for three basic Java computing technologies that are used often in the Java programming in this thesis. Java thread is used to perform a task in the parallel style. Understanding Java network programming is the basis of distributed object computing in Java. With a database involved in the applications, JDBC is often used to access the database.

2.3.1 Java Thread

A thread is a path of code execution through a program, and each thread has its own local variables, program counter and lifetime. If the underlying OS and the implementation of Java VM exploits the use of real multiple processors, multithread Java programs can achieve true simultaneous thread execution.

Threads can be created in two ways. One is to extend the *Thread* class, the other is to create the thread by implementing the *Runnable* interface. For example, the code fragment which spawns a new thread by extending the *Thread* class is like:

```
Public class NewThread extends Thread {
        Public void run() {    // … }
}
NewThread  nt = new NewThread();
nt.start();
```

The subclass *NewThread* consequently inherits the protected and public members from the *Thread* class. By invoking the *start()* method on the object *nt*, a new thread starts and invokes the run method of the *Thread* object. Meanwhile, the original thread is free to continue executing the statements that follow the *start()* call. The code fragment using the *Runnable* Interface is as following:

```
Public class MyClass extends SuperClass implements Runnable {
    Public void run() {   // …  }
}
Runnable r = new MyClass();
```

```
Thread t = new Thread(r);
t.start();
```

In this method, we also need to create an actual *Thread* object by passing the *Runnable MyClass* object reference to the constructor of the Thread. When *t.start()* is executed, the newly spawned thread begins execution by invoking the *run()* method of the *MyClass* object. This feature of execution is used often in the GUI programming and concurrent programming in the projects.

In order to ensure that the threads don't adversely affect one other, Java provides the thread concurrent control mechanism, such as the *synchronized* and *volatile* keywords, to control concurrent access to objects and variables. The *volatile* keyword is used to tell the VM that it should not keep a private copy of a variable and should instead interact directly with the shared copy. The *synchronized* modifier to a method declaration ensures that only one thread is allowed inside the method at a time, which is useful in the case that the state of an object is temporarily inconsistent. In Java, threads can be grouped together and associated with an instance of ThreadGroup. A thread group can be used to facilitate the management of threads. Thread groups allow the threads of the VM to be organized and can provide some inter-group security. Java also provides a thread scheduling mechanism to determine which thread is currently running on the processor and how long it is allowed to run before it is swapped off the processor to allow another thread to run.

2.3.2 Java Networking Programming

The *java.net* package provides an object-oriented framework for networking. The core of java networking support is the *Socket* and *DatagramSocket* classes. These classes define channels for communication between processes over an IP network. A new socket is created by specifying a host, either by name or with an *InetAddress* object, and a port number on the host. There are two basic kinds of network sockets on IP networks: sockets using TCP(Transport Control Protocol) and DatagramSocket using UDP (Unreliable Datagram Protocol). TCP is a reliable protocol in which data packets are guaranteed to be delivered correctly; while UDP makes no guarantee about the delivery of packets, or the order in which the packets are delivered. TCP sockets allow the user to treat a network connection as a stream; UDP doesn't allow this. Using UDP, the user always works with the individual datagram, which is an independent, self-contained message sent over the network and the arrival, arrival time and content are not guaranteed. The typical usage of *Socket* class and *DatagramSocket* class is like this:

Server-side	Client-side

ServerSocket s = new ServerSocket(5000);
// wait for a connection request from a client
Socket connect = s.accept();
InputStream in=connect.getInputStream();
OutputStream
out=connect.getOutputStream();
	InetAddress addr =
	InetAddress.getByName("rhost");
	Socket s = new Socket(addr, 5000);
	InputStream in = s.getInputStream();
	OuputStream out=s.getOuputStream();

Server-side	**Client-side**

DatagramSocket udps =
new DatagramSocket(5000);
byte[] dbf = {'H','I'};
InetAddress
ad=InetAddress.getByName("..");
DatagramPacket p = new
DatagramPacket(dbf,dbf.length,ad,5000);
udps.send(p);
	DatagramSocket udps =
	new datagramSocket(5000);
	byte[] dbf = new byte[1024];
	DatagramPacket p =
	new DatagramPacket(dbf,1024);
	udps.receive(p);

Table 2.3.1 Code fragments in Java network programming

2.3.3 Java Database Connectivity

Java Database Connectivity (JDBC) is the database connectivity package included in the core Java API. JDBC provides a database-independent interface for opening a connection to a relational database, issuing SQL calls to the database, and receiving a set of data as the result. JDBC acts as a Java implementation of the standard SQL call-level interface and is supported by most major relational database vendors. A JDBC driver

provides a bridge between the JDBC method calls and the native database interface. The architecture of JDBC is shown in Fig. 2.3.1, which illustrates the several ways that a JDBC driver can be configured to interact with an RDBS.

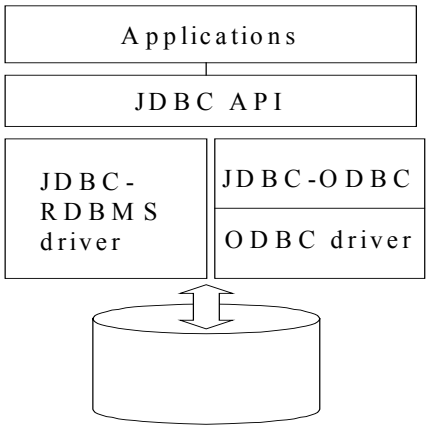


Fig. 2.3.1 JDBC driver configuration

The JDBC API offers *DriverManager*, *Connection*, *Statement* and *ResultSet* interfaces that mirror the basic concepts surrounding relational databases. The *DriverManager* class provides the means to load database drivers into a Java application or applet by searching a set of available drivers specified by the *sql.Driver*'s Java property. Once the necessary drivers have been loaded by the *DriverManager*, a connection to a database can be made by calling the *DriverManager*'s static *getConnection()* method. The desired database is specified with a String argument that acts as an URL-like address to the database. The *getConnection()* method on *DriveManager* either returns a *Connection* object that represents the connection to the named database, or throws an exception if the connection couldn't be established. The

Connection interface allows the user to create query statements to the database. *Query* statements are represented as *Statement* objects, such as *createStatement()*, *prepareStatement()*, and *prepareCall()*. The first method is used for simple SQL statements that don't involve any parameters. An SQL statement involving input parameters or for multiple execution times can be created using the *prepareStatement()* method, which returns a *PreparedStatement* object. A stored SQL procedure can be accessed through an SQL statement created through the *prepareCall()* method on a connection object, which returns a *CallableStatement* object. Rows of data returns from the execution of a statement against a database are represented as *ResultSet* objects in JDBC. A *ResultSet* object provides ways to iterate through the rows of data returns as the result of an SQL query, through its *next()* method and data fields within each row can be retrieved by name or by column index number using its get methods.

Projects implemented Using Java technology

The above three technologies are used in all projects in this thesis. We distinguish the projects by using Java threads, or JDBC or networking according to the dominant technology applied in the project.

Project ECGEN, which uses Java threads more often, generates all possible electron configurations if given the total electron number and some restrictions for the electron orbitals. The solution for this problem is like solving linear multiple variable equations with constraints. To operate this program, some senses of generating electron configurations are needed. The program first finds all possible electron distributions on

the N level shells, computing threads are then created for each possible distribution. All these computing threads start to do the calculation simultaneously. A master thread waits to assemble the final electron configurations according to all possible configurations.

Project OPAVIEWER is used to view the EOS and opacity data calculated by EOSOPA. The server-side process sends the requested data to the client, and the client renders the data in a graph using Java 2D.

2.4 Atomic Web Database Using Oracle DBMS

The core of this thesis is to set up an atomic database, which contains most of the data needed by the ICF applications, such as the atomic data, EOS and opacities. In this section, we first give an introduction of the architecture of the Oracle database management system, which is extensively used in this thesis. Then, we discuss the atomic web database using Oracle DBMS. Section 2.4.2 shows the Oracle network computing architecture, which provides a powerful tool for the application deployment for all projects in this thesis.

2.4.1 Oracle Database Architecture

An Oracle database [WORC] separates the physical structure from the logical structure such that the physical storage of data can be managed without affecting the access to logical storage structures. The Oracle database's physical structure consists of three

types of files: one or more datafiles, two or more redo log files and one or more control files. These files provide the actual physical storage for database information. The data file contains tables, indexes, clusters, sequences, data dictionary and so on. The control file contains information about the database's physical structure and status. It has information about the total number of data files, log files, redo log members, name and location of each data file, etc. Oracle records all changes in the redo log file and uses it to regenerate the transaction changes in case of failure. An Oracle instance writes to redo log group in cyclical order. The Oracle database's logical structure consists of one or more tablespaces and the database's schema objects. Tablespaces are logical storages that group related logical structures. A schema is a collection of database objects, which are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters and database links. Oracle also allows fine-grained control of disk space use through the logical structures, including blocks, extents and segments.

An Oracle server uses memory structures and processes to manage and access the database. To enable efficient data manipulation and communication among the various processes, Oracle uses a shared global area (SGA). The SGA is a shared memory region that contains data and control information for one Oracle instance. An Oracle instance consists of an SGA and the Oracle background processes. Each instance has its own system global area and Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. When a server process is started Oracle also

creates a program global area which is a memory buffer that contains data and control information for a server process. An Oracle server has two general types of processes: user processes and Oracle processes. User processes are created and maintained to execute the software code of an application program and communicate with the server processes through the program interface. Oracle processes include server processes and background processes. A server process is in charge of communicating with the user process and interacting with Oracle to carry out requests of the associated user process. A set of background processes also is created for each instance. The background processes, such as Database Writer, Log Writer, Checkpoint, System Monitor, Process Monitor, Archiver, Recoverer, Dispatcher, Lock, Job Queue and Queue Monitor, asynchronously perform I/O and monitor other Oracle processes to provide parallelism for better performance and reliability.

Implementation of atomic database

The atomic database stores the UTA-based atomic data for UTAOPA model and RSSOPA model. It also stores EOS data and opacities on a predefined temperature and density grid. According to the complexity of the atomic data structure, we store the data either in the pure relational model or in the object-relational model.

An entity OPERATION represents a calculation process. It records the author of the calculation, the model used in the calculation and the elements used. The SQL language used to create a schema of OPERATION table is as following:

```
Create table OPERATION    (
```

```
 OID number(5) CONSTRAINT Operation_OID_pk PRIMARY KEY,
Element      ELEMENT_OBJTYP,      Model      varchar2(10),
Author      varchar2(30),                  Version      varchar2(10),
Operation_date      DATE   ).
```

For UTAOPA and RSSOPA models, there are three basic tables: one for storing the photoionization cross sections, one for the photoexcitation and one for the average configuration energies. Because UTAOPA uses LS coupling while RSSOPA uses JJ coupling, the object types for orbitals are different. Also because the strategies used in RSSOPA are different from those used in UTAOPA (while the RSSOPA model explicitly calculates the transition width for every transition line and therefore the transition width can be stored in the database, the UTAOPA model embeds the transition width calculation in the opacity calculation and therefore the transition width can not be stored in the database), the data structures are slightly different. More details on these tables can be found in Appendix A. These tables are used in Project JEOSOPA and SPECTRA.

For EOS data, the element name, the temperature ID, the density ID and the calculation model construct the primary key to uniquely identify each record. All EOS data, such as electron pressures, ion pressures and total energies are stored in the single table EOSTABLE. These tables are used in Project OPAVIEWER and MIXOPA.

For opacity data, we use the element name, the temperature ID, the density ID and the calculation model as the primary key. Additionally, we need another attribute to describe the photon energy grid. These tables are used in Project MIXOPA.

2.4.2 Oracle Network Computing Architecture

Oracle possesses a very good position in the competition of the network computing world. Its vision for the infrastructure is based on the premise that no single technology or standard will win in the foreseeable future, and therefore Oracle's strategy is to support them all. Oracle achieves this based on its CORBA-based Network Computing Architecture (NCA).

NCA is the cross-platform infrastructure for developing and deploying object-based, network-centric applications in an open, heterogeneous environment. As shown in Fig. 2.4.1, the core of the architecture are two standards: CORBA and HTTP. CORBA provides a distributed object computing environment, includes IIOP for object interoperability and IDL for language-neutral interfaces. With the addition of HTTP transaction services, Oracle provides a robust web environment.

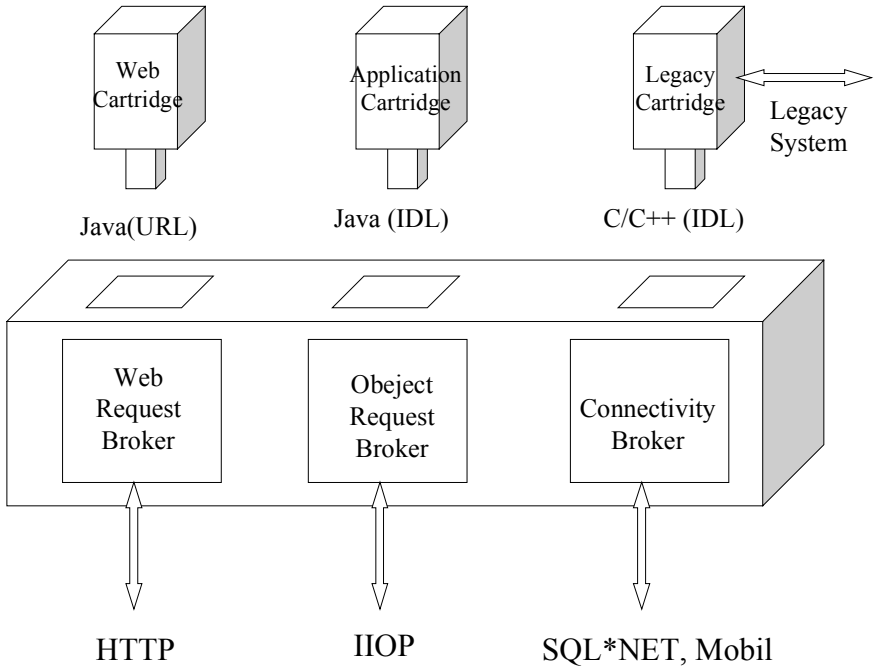


Fig. 2.4.1 Architecture of Oracle Universal Application Server in NCA

The NCA architecture consists of pluggable objects called cartridges, a software bus called Inter-Cartridge Exchange (ICX) and extensible clients, application servers and database servers. A cartridge is a manageable object that uses an IDL to identify itself to other objects in the distributed system. Cartridges have access to Universal Cartridge services such as registration service, instantiation service, invocation service, security service and others. ICX enables cartridges distributed across a network to communicate with each other using IIOP and HTTP protocols. The Universal Application Server (as shown in Fig. 2.4.1) plays a center role in the NCA. It acts as a platform for application logic and make the NCA client side thin, and the applications centric manageable. Oracle Universal Server provides robust, scalable data storage. By

adding procedures to the data stored in the database, Oracle data server provides significant performance and management advantages in many applications, and with object-relational database technology, it can create new data types with sophisticated functionality.

Application deployment under NCA architecture

In order to make our application available and accessible from the Internet, we need an application server to provide these services. In this thesis, we deploy our applications under NCA architecture. Fig. 2.4.2 displays the logic architecture of the atomic database computing system.

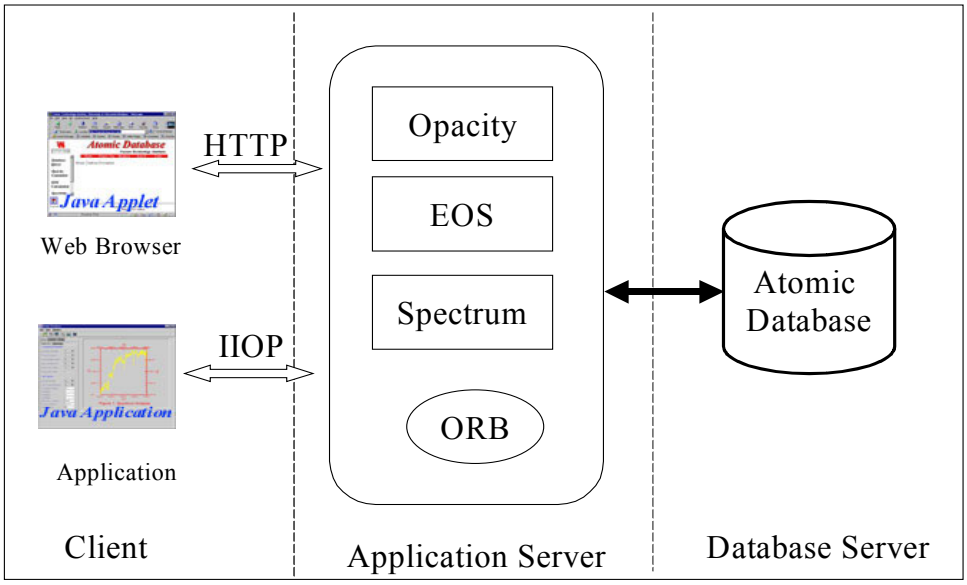


Fig.2.4.2 The logic architecture of the atomic database computing system

As shown in the figure, there are two kinds of clients: one connecting the application servers through HTTP protocol and the other through IIOP protocol. The application logic such as calculating the EOS opacities and spectrum analysis is on the application server and connects with the atomic database through JDBC.

2.5 Summary of Information Technologies Used for Implementation

As indicated by its title, this thesis has four parts. The first part is the high performance computation of atomic data. Besides the original ATBASE model, we develop a new atomic model - RSSOPA to calculate JJ coupling atomic data based on the UTA method. MPI is used in the parallel computing environment to speed up the calculation since enormous number of transitions are involved. The second part is to create a database to store the data such as energy levels, EOS and opacities needed by the ICF applications. The third part is to develop applications that allow users to generate data such as EOS, mixed opacities and spectrum for ICF applications based on the database. The fourth part is the graphic user interface, which provides users an easy way to interact with the program. The following figure illustrates the four parts in the distributed atomic data computing system.

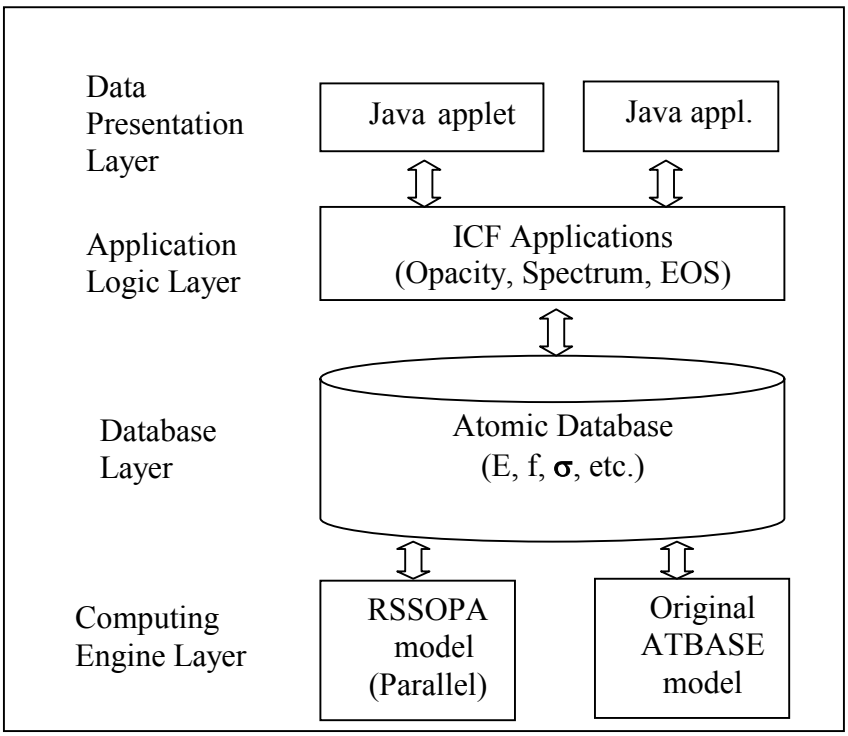


Fig. 2.5.1 Four Parts in the Distributed Atomic Data Computing System

In order to integrate these four parts and make them accessible from the Internet, the CORBA architecture and some other frameworks are also used. According to the functionalities they provide, this thesis breaks up into seven projects, which are shown in the list of projects at the beginning of the thesis.

The Oracle DBMS is used to manage the atomic data. Depending on the complexity of the atomic data structure, we store the data either in the relational model or in the object-relational model.

There is no absolute reason to choose a certain technology. Each technology has it own strength but also has its weakness. For example, CORBA provides multi-

programming language support, but it is more difficult to program than EJB. We choose CORBA for Project JEOSOPA and SPECTRA because we need to interact with some existing programs such as UTAOPA and modules written in FORTRAN. We choose EJB for Project MIXOPA because we wrote a pure Java program to do the calculation. EJB makes it much easier to interact with the database and to deploy onto multi-tier architecture. Technologies such as Java thread, networking and JDBC are basic to writing efficient Java software.

Chapter 3

Atomic Models and Opacity Calculations

In this chapter, we delve into the physics that provides the atomic data based on the specific code. In this thesis, several existing codes are used. ATBASE is used for the calculation of atomic data under both DTA and UTA assumptions. The DTA part of ATBASE is based on Cowan's code, which calculates the non-relativistic atomic structure. The UTA part of ATBASE calculates quantities for the average configuration and uses the non-relativistic forms of UTA moments. We discuss the atomic models used in ATBASE in Section 3.1.1. Another existing code we use to calculate EOS and opacities is EOSOPA, which consist of two models DTAOPA and UTAOPA. DTAOPA uses the DTA atomic data generated by ATBASE, while UTAOPA uses the UTA atomic data. Because no collisional or recombination rate coefficients can be currently generated under the average configuration approximation, UTAOPA can only run under the LTE model. DTAOPA can also calculate non-LTE opacities since a lot of atomic processes such as dielectronic recombination, election collision ionization, can be

obtained from ATBASE under the DTA method. The various atomic processes and plasma models used in the code are discussed in Section 3.1.2 and 3.2.1. In addition to the existing codes, we developed a new model, RSSOPA, which is based on the relativistic average configuration approximation and UTA method. It uses JJ coupling for the atomic data. This model provides more accurate spectrum data for high-Z elements. We discuss this model in Section 3.2.2.

3.1 Atomic Models

3.1.1 Non-relativistic Theory of Atomic Structure

The atomic structure of a many-electron system is determined by the solution of a partial differential equation (called the Schrödinger wave equation),

(1) $$(H - E)\psi = 0,$$

where H is the Hamiltonian operator for the system and E is the total energy. The operator H depends on the system such as atomic, molecular or solid-state systems and the quantum mechanical formalism such as non-relativistic and Dirac-Coulomb. The non-relativistic calculation of atomic structure is based on the non-relativistic Hamiltonian operator [COW81][FS86]:

(2) $$H = -\frac{1}{2}\sum_{i=1}^{N}(\nabla_i^2 + \frac{2Z}{r_i}) + \sum_{ij}\frac{1}{r_{ij}} + \frac{1}{2}\sum_i \zeta_i(r_i) l_i \cdot s_i,$$

where N is the number of electrons and Z is the nuclear charge of the atom, r_i is the distance of the i^{th} electron from the nucleus, and r_{ij} is the distance between electron i and electron j. The first term of the Hamiltonian represents the kinetic energy of the electrons and the Coulomb energy between the electrons and the nucleus. The second term represents the electrostatic Coulomb interaction among the electrons. The final term of the Hamiltonian represents the magnetic interaction energy between the spin of the electrons and their own orbital motion.

To obtain the eigenvalue and eignenfunction of the Schrödinger equation, approximations must be used because of the high dimensionality of the equation. The usual approach is to use perturbation theory. First, the one-electron wavefunctions are constructed using the central-field model. In the central-field model, one electron is assumed to move in a time-average electro-magnetic field which is generated by the nucleus and the other electrons and therefore is spherically symmetric. So the wave function of the electron is a separable function of only (r,θ,ϕ) in the form:

$$\varphi_i(r) = \frac{1}{r} P_{n_i l_i}(r_i) Y_{l_i m_i}(\theta_i, \phi_i) \sigma_{m_{si}}(s_{iz}). \tag{3}$$

Here P(r) is the radial function, Y(θ,ϕ) is the spherical harmonic function and σ(s) is the spin function. Secondly, we need to construct the basis function for the entire atom from the one-electron orbitals $\varphi_i(r)$, which should reflect the physical indistinguishable property of electrons. The antisymmetrized configuration state function is the linear combination of uncoupled functions, which can be represented as a determinant:

(4) $$\Phi = (N!)^{-1/2} \sum_{P} (-1)^{P} \varphi_1(r_1)\varphi_2(r_2)...\varphi_N(r_N).$$

Finally, the configuration interaction wave function for the states of the entire atom is written as an expansion of the antisymmetrized configuration functions Φ.

(5) $$\psi = \sum_j c_j \Phi_j$$

From the eigenequation, the total energy of the atom is given by

(6) $$E = <\psi | H | \psi>,$$

if $<\psi | \psi> = 1$. Using the multipole expansion for $1/r_{ij}$,

(7) $$\frac{1}{r_{ij}} = \sum_k \frac{r_<^k}{r_>^{k+1}} P^k(\cos\theta),$$

where $r_<, r_>$ are the lesser and greater of r_i and r_j, respectively, and $P^k(\cos\theta)$ is a Lengendre polynomial in $\cos\theta$ where θ is the angle between r_i and r_j, the energy can be written as

$$E = \sum_{ij} c_i c_j H_{ij}$$

$$= \sum_{ij} c_i c_j \left(\sum_{stuv;k} A^{ij}_{stuv;k} R^k(s,t;u,v) - \frac{1}{2} \sum_{qw} C^{ij}_{qw} L_{qw} \right)$$

$$= \sum_{stuv;k} A_{stuv;k} R^k(s,t;u,v) - \frac{1}{2} \sum_{qw} C_{qw} L_{qw}$$

where R^k is the two-dimensional integral,

$$(8)\quad R^{k}(s,t;u,v) = \int_0^{\infty}\int_0^{\infty} dr dr' P_s(r) P_t(r') \frac{r_<^k}{r_>^{k+1}} P_u(r) P_v(r'),$$

and L_{qw} is the one-dimensional integral,

$$(9)\quad L_{qw} = \int_0^{\infty} dr P_q(r) \left[\frac{d^2}{dr^2} + \frac{2Z}{r} - \frac{l(l+1)}{r^2} \right] P_w(r).$$

The $A^{ij}_{stuv;k}$ and C^{ij}_{qw} are called angular coefficients and can be computed using Racah algebra. According to the variational principle, the total energy of the atom is an eigenvalue of the interaction matrix, and the expansion coefficients of the wave function are the corresponding eigenvector. In order to compute the interaction matrix, the radial functions need to be computed, which should be chosen to minimize the center-of-gravity energy for a configuration. Through the derivation using the variation method with the orthonormalization conditions,

$$(10)\qquad \int_0^{\infty} P^{*}_{n_i l_i}(r_1) P_{n_j l_j}(r_1) dr_1 = \delta_{n_i n_j}$$

the Hartree-Fock equation for the radial wave function is expressed as

$$(11)\quad \left[-\frac{d^2}{dr^2} + \frac{l_i(l_i+1)}{r^2} - \frac{2Z}{r} + \sum_{j=1}^{q}(w_j - \delta_{ij}) \int_0^{\infty} \frac{2}{r_>} P_j^2(r_2) dr_2 - (w_i - 1) A_i(r) \right] P_i(r)$$
$$= \varepsilon_i P_i(r) + \sum_{j(\neq i)=1}^{q} w_j \left[\delta_{lilj} \varepsilon_{ij} + B_{ij}(r) \right] P_j(r),$$

where

$$(12)\quad A_i(r) = \frac{2l_i + 1}{4l_j + 1} \sum_{k>0} \begin{pmatrix} l_i & k & l_j \\ 0 & 0 & 0 \end{pmatrix}^2 \int_0^{\infty} \frac{2 r_<^k}{r_>^{k+1}} P_i^2(r_2) dr_2$$

and

$$(13)\quad B_{ij}(r) = \frac{1}{2}\sum_k \begin{pmatrix} l_i & k & l_j \\ 0 & 0 & 0 \end{pmatrix}^2 \int_0^\infty \frac{2r_<^k}{r_>^{k+1}} P_j(r_2) P_i(r_2) dr_2 .$$

The first three terms arise from the kinetic energy and the nuclear potential energy. The fourth term comes from the direct portion of the electron-electron interactions E^{ij} for electrons both equivalent and non-equivalent to i, which has physical meaning as the potential energy of the i electron in the averaged field of the other N-1 electrons. The fifth term arises from the exchange portion of the interaction energy between the electron i and the other equivalent $(w_i - 1)$ electrons, which is the correction corresponding to the partial positional correlation among electrons of parallel spin. The final term in the above equation arises from the orthogonality requirement. This equation is generally solved using the self-consistent-field (SCF) method. Because of the complication involved in solving the HF equations, the local potential approximation is often used. In this approximation, the Schrödinger equation becomes:

$$(14)\qquad \left[-\frac{d^2}{dr^2} + \frac{l_i(l_i+1)}{r^2} + V^i(r) \right] P_i(r) = \varepsilon_i P_i(r),$$

where $V^i(r)$ is the potential function that the electron i moves in, for which there are several kinds of potential forms:

The Hartree-Fock-Slater potential (HFS)

$$(15)\qquad V^i(r) = -\frac{2Z}{r} + \sum_{j=1}^{q} w_j \int_0^\infty \frac{2}{r_>} P_j^2(r_2) dr_2 - \frac{3}{2}\left(\frac{24\rho}{\pi} \right)^{1/3},$$

where ρ is the total spherically averaged electron density.

The Hartree-Statistical-Exchange Potential (HX)

$$V^i(r) = -\frac{2Z}{r} + \sum_{j=1}^{q}(w_j - \delta_{ij})\int_0^\infty \frac{2}{r_>}P_j^2(r_2)dr_2 - k_x f(r) f(\rho)\left(\frac{24\rho}{\pi}\right)^{1/3}, \tag{16}$$

which considers the self-energy effect and adds a statistical exchange energy correction term.

The Hartree-Slater Potential (HS)

$$V^i(r) = -\frac{2Z}{r} + \sum_{j=1}^{q}(w_j - \delta_{ij})\int_0^\infty \frac{2}{r_>}P_j^2(r_2)dr_2 - \left(\frac{24}{\pi}\right)^{1/3}\left[\rho_s^{1/3} - (2\rho)^{1/3}\right], \tag{17}$$

which is similar to the HX potential but uses a different exchange energy expression.

After we obtain the radial portion of the wave function and apply the Racah algebra, we can calculate the energy level structure of the atom. Depending on what wave functions are used as the basis function in the Hilbert space, we may have the energy level structure under the single configuration approximation or the interaction configuration approximation.

Under the single configuration approximation, the set of basis functions is constructed according to the general electronic configuration form:

$$\prod_{j=1}^{q}\left(n_j l_j\right)^{w_j}.$$

The energy matrix $\left\langle \psi_b \mid H \mid \psi_{b'} \right\rangle$ consists of two terms: one term involving one-electron operator, such as the kinetic energy and the electron-nuclear energy, which is

easy to compute, and an another term involving two-electron operator, that is the electron-electron Coulomb energy, which can be expressed as follows:

$$\left\langle \psi_b \mid \sum_{i<j} g_{ij} \mid \psi_{b'} \right\rangle = \sum_{j=1}^{q} \frac{w_j (w_j - 1)}{2} \left\langle \psi_b \mid g_{ij} \mid \psi_{b'} \right\rangle +$$

$$\sum_{i<j}^{q} w_i w_j \left[\left\langle \psi_b \mid g_{ij} \mid \psi_{b'} \right\rangle - \left\langle \psi_b \mid g_{ij} \mid \psi_{b'}^{ex} \right\rangle \right],$$

where g_{ij} is the electron-electron Coulomb energy:

$$\frac{2}{r_{mn}} = \sum_{0}^{\infty} \frac{2 r_{<}^{k}}{r_{>}^{k+1}} C_m^k \cdot C_n^k. \tag{18}$$

Applying the Wigner-Eckart Theorem, the matrix element can be written as a product of a term purely dependent on the angular momentum and a term dependent on the radial position. After obtaining the matrix elements, we can solve the matrix eigenvalue problem using some standard methods.

In order to consider the configuration interaction effect, we need to expand the wave function of the atom to include other sets of basis functions belonging to their corresponding configurations. The similar techniques to the single-configuration method are used to construct the energy matrix for the configuration interaction.

3.1.2 Atomic Radiative and Collision Processes

In this section, we discuss the atomic radiative and collision processes that are considered in the code ATBASE. According to the initial and final states, the atomic radiative processes can be classified into three types; that is, bound-bound transition (excitation), bound-free transition (photoionization) and free-free transition (Bremsstrahlung). For collisions, the two most important processes are electron collisional excitation and ionization. These processes and the formulas that are used by ATBASE are given in the following sections.

3.1.2.1 Atomic Radiative Processes

1. Bound-Bound transition

The bound-bound transition occurs between two bound states. In the quantum mechanical theory, the probability per unit time of an atom in a specific state j (γ′J′M′) making a transition to a state i (γJM) is

$$(19)\quad a_{ji} = \frac{64\pi^4 e^2 a_0^2 \Delta E^3}{3h^4 c^3} \sum_q \left| \left\langle \gamma JM \left| \sum_{i=1}^{N} r_i C_q^{(1)}(i) \right| \gamma' J' M' \right\rangle \right|^2,$$

under the electric dipole approximation. Here q is the polarization direction $(0, \pm 1)$. Applying the Wigner-Eckart theorem, the total transition probability from a state γ′J′M′ to all states of the level γJ is written as

$$(20)\quad gA = \frac{64\pi^4 e^2 a_0^2 \Delta E^3}{3h^4 c^3} S,$$

where $S \equiv |< \gamma J \| \sum_{i=1}^{N} r_i C^{(1)}(i) \| \gamma' J' >|^2$, which is called the line strength. The oscillator strength is defined by

$$f_{ij} = \frac{\Delta E}{3(2J+1)} S, \tag{21}$$

which has the physical meaning of the total probability of absorption from a specific lower level i to all (2J′+1) states of the upper level j.

Under the single-electron transition approximation, the oscillator strength is reduced to

$$f_{ij} = \frac{\Delta E}{3(2J+1)} (-1)^{2l+2j'+1} (2j+1)(2j'+1) \begin{Bmatrix} l & s & j \\ j' & 1 & l' \end{Bmatrix}^2 P^2{}_{nl,n'l'} \tag{22}$$

where

$$P^2_{nl,n'l'} = (2l+1)(2l'+1) \begin{pmatrix} l & 1 & l' \\ 0 & 0 & 0 \end{pmatrix}^2 | \int_0^\infty P_{nl}(r) r P_{n'l'}(r) dr |^2 . \tag{23}$$

The UTA oscillator strength is calculated using Equation(22) in ATBASE. For the general condition which involves multiple occupied outer subshells, Racah algebra is needed to handle the angular momentum coupling in the calculation of the transition matrix elements.

2. Bound-Free Transition

The bound-free transition is similar to the bound-bound transition except that the final state is a continuum state. The numeric calculation of the continuum wave

functions needs more care than calculations of bound-state wave functions because of the conditions of the orthogonality and the asymptoticity that the continuum wave function should satisfy. The transition probability for the photon ionization is usually expressed in terms of a photoionization cross section:

$$Q_{ij} = \frac{4\pi^2 e^2 a_0^2}{\hbar c} \frac{df_{ij}}{d\varepsilon}, \tag{24}$$

where $df_{ij} / d\varepsilon$ is called the oscillator strength density. Under the single-configuration approximation, the calculation of the oscillator strength density is the same as the calculation of bound-bound oscillator strength, except that the final wave function is replaced by the continuum wave function in the radial dipole reduced matrix element. For highly ionized atoms, configuration-interaction effects between the bound and continuum states are very small because the continuum states are well separated from the bound states. In the case of discrete bounded states lying within the energy range of the continuum states, the configuration interaction effect should be considered. If the discrete state has a perturbation from the continuum, the discrete state may autoionize and spread out into a resonance line shape (Fano profile [FN65]) with half-width at half-maximum $\Gamma_a = 0.5\hbar A^a$, where A^a is the autoionization transition probability rate.

3. Free-Free Transition

The high energy incident electron is decelerated by the Coulomb field of the scattering atom while a photon is emitted. This process is called Bremsstrahlung. The initial incoming electron wave interacts with the Coulomb field of the scattering atom

and with the electromagnetic field of the emitted photon. The transition matrix element M_{fi} between the initial state ψ_i and the final state ψ_f is

$$M_{fi} = \int \psi_f^* \varepsilon \cdot p e^{-iq \cdot r} \psi_i d\tau,$$

where $\varepsilon \cdot p e^{-iq \cdot r}$ describes the interaction of the electron with the radiation field. ε and p are the photon polarization and wave vector, respectively. $\psi_{f,i}$ are the initial and final wave functions which are in the form:

(25) $$\psi_k = \frac{1}{2k} \sum_{i=0}^{\infty} i^l (2l+1) e^{\pm i\delta l} R_{kl}(r) P_l(\cos\theta),$$

which represent the solution of the Lippman-Schwinger equation in the spherically symmetric potential. The radial wave function $R_{kl}(r)$ is the solution of the radial Schrödinger equation corresponding to the energy $E = \frac{\hbar^2 k^2}{2m}$.

The Bremsstrahlung process contributes to the continuous spectrum. In ATBASE, it is calculated using the Kramers formula with the Gaunt factor correction,

(26) $$\sigma^{ff} = \sigma_K^{ff} g_{ff},$$

where σ_K^{ff} represents the Kramers cross section and g_{ff} represents the Gaunt factor.

3.1.2.2 Atomic Collision Processes

In a hot dense plasma, there are many collision processes. The electron-ion collision is the most important process that contributes to the transition arrays and the redistribution

of the charge states. Depending on the final status of the ion and the electron, the collision processes are classified into electron collision excitation (deexcitation) and electron collision ionization (three body recombination). In the electron collision excitation, a free electron that moves near an ion loses energy by inducing a transition of a bound electron from a lower state into a higher state; while in the electron collision ionization, a bound electron is knocked out into the continuum state by a free electron.

Theoretical calculations for the collision process need approximations [BR83]. If the relative velocity of the colliding electron is much higher that the velocity of the optical electron, the Born approximation can be used, which is the first order of the perturbation theory. In this approximation, the motion of the incident and outgoing electron are described using the plane waves. In the case of $qr << 1$, the Born cross section can be reduced to the Bethe formula:

$$\sigma = \frac{8\pi a_0^2 f_{ji}}{E\Delta\varepsilon} \ln \frac{q_0}{k_0 - k_1}, \tag{27}$$

where f_{ji} is the oscillator strength. However, the Born approximation does not include the exchange of the incident and atomic electrons. The effect of the electron exchange is taken account in the Born-Oppenheimer approximation using antisymmetrical wave functions.

In ATBASE, the distorted-wave-exchange (DWE) method is used to calculate the election collision cross sections. The DWE method takes into account the distortion of the wave functions by the mean field of the atom or ion. The mean field consists of

the long range Coulomb field $U(r) = -\frac{z-1}{r}$ and a short range attractive part . The short range attraction leads to an increase of the cross section because of the closer distance. The formula of collision strength in DWE approximation can be found in reference [PW93].

3.1.2.3 Resonant Processes

The processes of dielectronic recombination (DR), Auger ionization (AI) and resonant excitation (RE) and ionization (RI) all are based on the reaction of electron capture into nl-states of an ion X_z:

$$X_{z+1}(\alpha) + e \rightarrow X_z^{**}(\gamma), \qquad \gamma = nlLSJ$$

and then have the second decay stage, which can be the Auger decay emitting an electron:

$$X_z^{**}(\gamma) \rightarrow X_{z+1}(\alpha') + e, \quad \text{(AI,RE)}$$

$$X_z^{**}(\gamma) \rightarrow X_{z+2}(\alpha') + 2e, \quad \text{(RI)}$$

or the radiative transition emitting a photon and transfer to a stable state which is below the ionization limit:

$$X_z^{**}(\gamma) \rightarrow X_z(\gamma') + h\nu. \quad \text{(DR)}$$

The capture is possible only within a narrow interval of the incident electron energy around the value $\varepsilon(\gamma,\alpha) = E_\gamma - E_\alpha$. The width of the energy interval is equal to the level width δ_γ which is defined by the total decay probability

$$\delta_\gamma = \hbar(W(\gamma) + A(\gamma),$$

where W and A are the Auger and radiative decay probabilities. The capture cross section is given by the dispersion formula

$$\sigma_c(\gamma \mid \varepsilon) = \sigma_c(\gamma)\frac{\delta_\gamma / 2\pi}{\left[\varepsilon - \varepsilon(\gamma,\alpha)\right]^2 + \delta_\gamma^2 / 4}, \tag{28}$$

where $\sigma_c(\gamma)$ depends on the energy $\varepsilon(\gamma,\alpha)$ of the incident electron. The total capture rate is

$$\kappa_c(\gamma) = F(\varepsilon)\upsilon\int_0^\infty \sigma_c(\gamma \mid \varepsilon')d\varepsilon', \tag{29}$$

where $F(\varepsilon)$ is the Maxwellian distribution of electrons.

3.2 Opacity Calculations

Theoretical calculation and experimental determination of hot dense plasma opacities have long been of interest from astrophysics to inertial confinement fusion (ICF) research. Modeling of the energy transport in hot dense plasmas relies on radiative opacities. Several opacity models have been applied to calculate the plasma opacities, such as the detailed term accounting (DTA) method, the detailed configuration

accounting (DCA) method, the unresolved transition array (UTA) method and some average-atom models based on the statistical theory. These models need validation by experiments. However, high quality experimental measurements of the x-ray opacities of highly ionized materials are difficult since the experiment errors must be accurately specified and the plasma condition must be precisely determined. Such measurements became possible by the technique of radiative heating using the intense x-rays emitted by laser-irradiated targets. The point projection spectroscopy technique has been extensively applied for this kind of experiment. The plasma to be studied is created either by direct or indirect irradiation. An auxiliary plasma, whose dimensions are small compared to the main plasma expansion, is generated by a synchronized laser and generates an x-ray source which probes the main plasma. The attenuation of the x-ray probe through the expanding plasma is measured with a space resolution on the order of the point source diameter and with a time resolution of the duration of the auxiliary source plasma.

The theoretical aspect of opacity research [RS92][BOG89][BOS95] is also very complex and necessarily uses approximations. The calculation of opacity needs two components: the atomic radiative quantities determined by the atomic model and the populations calculated by the radiative dynamics. The radiative processes, as discussed in Section 3.1, mainly involved three parts: photoexcitation, photoionization and bremsstrahlung. The atomic theory based on the quantum mechanics has been solidly established to calculate these processes and several freely-distributed codes are

available, such as Cowan's code and Grant code. However, these calculations are largely based on the free atom assumption, that is, they assume the atom is isolated and there are no interactions between this atom and the environment. This approximation is appropriate in some plasma condition ranges but does not hold when the plasma is strongly coupled, in which the Coulomb interaction energies are equal or greater than the average kinetic energy of the plasma particles, i.e. $\Gamma = (ze)^2 akT \geq 1$, where $a = (3N/4\pi)^{-1/3}$ is the inter ion distance. If the plasma environment is considered in the determination of the atomic structure and transition properties, it is expected that different behavior will be exhibited, such as formation of energy bands, the shift of energy levels and pressure ionization. It is more difficult to coherently handle the problem as a whole rather than single out the atom but include the environment condition as the additional correlative energy and exchanged energy, which is derived from the statistical theory (Density Function Theory) and perturbation analysis. Therefore, the calculation of atomic structure in plasmas is basically the same as the calculations of free atom except that the modified potential reflects the plasma environment effects.

In the following sections, we give the three different plasma models that are used in the code EOSOPA. Because ATBASE currently doesn't include the calculation of all of the rate coefficients under the average configuration assumption, UTAOPA can only run the LTE model.

3.2.1 Plasma Models and Various Rate Coefficients

3.2.1.1 Local Thermodynamic Equilibrium (LTE)

LTE occurs in plasmas whose dimensions are significantly smaller than the mean free path of the photons emitted from the plasma, but are much longer than the collision length of the electrons and the ions. The photons may either escape from the plasma or be reabsorbed in some other part. The electrons and ions are colliding at a high rate and their distributions of velocities and excited states are in equilibrium. The population N_i is given by the Saha equation [ST64]:

(30) $$\frac{N_{i+1}N_e}{N_i} = \frac{Z_e Z_{i+1}}{Z_i} e^{-\Phi_i / kT},$$

where N_e is the number of free electrons, Φ_i is the ionization potential of ion i ,

$$Z_e = 2\left(\frac{2\pi m_e kT}{h^2}\right)^{3/2},$$

$$Z_i = \sum_m g_{im} e^{-E_{im}/kT}.$$

The Saha equation is solved with the constraint of particle conservation

(31) $$N = \sum_{im} N_{im}$$

and charge conservation

(32) $$N_e = \sum_{im} q_i N_{im}$$

where N is the specified total particle density, and q is the charge of ion i. The population distribution of the electrons in the various excited states is given by the Boltzmann distribution

(33) $$N_{il} = g_{il}(N_i / Z_i)e^{-E_{il}/kT}.$$

3.2.1.2 Coronal Equilibrium (CE)

The other extreme is the very low density and optically thin plasma range. Such plasmas occur frequently under both astrophysical and laboratory conditions. Under the CE condition, the upward excitation rate by collisions is so low relative to the spontaneous decay that an electron excited to an upper level will most likely decay to the ground state before experiencing a second excitation. Moreover, in low density optically thin plasmas the photoionization and photoexcitation processes have very low rates. The dominant processes are electron impact ionization and radiative and dielectronic recombinations. The charge state distribution is calculated by equating the rates of these processes,

(34) $$n_e N_{i-1} I_{i-1,i} = n_e N_i R^{(2)}_{i,i-1},$$

(35) $$\frac{N_i}{N_{i-1}} = \frac{I_{i-1,i}(T_e)}{R^{(r)}_{i,i-1}(T_e) + R^{(d)}_{i,i-1}(T_e)}.$$

Because all ion stages are assumed in the ground state in CE, this equation is a relation between the partial densities of the ground states of two adjacent charge states.

3.2.1.3 Collisional Radiative Equilibrium (CRE) Steady State

The CRE model is an intermediate model between LTE and CE. It tends to the CE in the low density limit, and to LTE for high density plasma. Consider a plasma of atomic number Z, electron temperature T_e and ion density n_i. The densities of the charge state i and of the excited state j are N_i, N_{ij}, respectively. The change rate of the population of a particular level is affected by several processes and their inverse processes as listed in Table 3.2.1.

The rate equation for the CRE steady state model can be written as

$$N_i \sum_{j \neq i}^{N} W_{ij} = \sum_{j \neq i}^{N} N_j W_{ji}, \tag{36}$$

where W_{ij} represents the summation of the upward transition processes: stimulated absorption, collision excitation, photoionization, collision ionization; while W_{ji} represents the summation of the downward transition processes: spontaneous and stimulated emission, collision deexcitation, rediative and dielectronic recombination, and three-body collision recombination. To calculate accurate rate coefficients for all kinds of these processes is very difficult. In practice, the empirical formulas are often used. Reference [PW93] givens these formulas used in ATBASE.

Table 3.2.1 Atomic Processes involved in the CRE model

Reaction: $N_i^m \Leftrightarrow N_i^{m'} + h\nu$			
(a)	(b)	(c)	(b)
Spontaneous Decay	$N_i^m A_i^{mm'}$	Stimulated Absorption	$n_\gamma N_\gamma^{m'} B_i^{m'm} \Delta h\nu$
Reaction: $N_i^m + e \Leftrightarrow N_{i+1}^m + e + e$			
(a)	(b)	(c)	(b)
Electron collisional Ionization	$n_e N_i^m I_{i,i+1}^m$	3-Body recombination	$n_e^2 N_{i+1} R_{i,i+1}^{(3)m}$
Reaction: $N_i^m + e \Leftrightarrow N_i^{m'} + e$			
(a)	(b)	(c)	(b)
Electron collisional excitation	$n_e N_i^m E_i^{mm'}$	Electron collisional Deexcitation	$n_e N_i^{m'} D_i^{m'm}$
Reaction: $N_i^m + h\nu \Leftrightarrow N_{i+1}^{m'} + e$			
(a)	(b)	(c)	(b)
Photoionzation	$n_\gamma N_\gamma^m \beta_{i,i+1}^{m'm}$	Radiative recombination	$n_e N_{i+1} R_{i,i+1}^{(r)m}$
Reaction: $N_i^{mm'} \Leftrightarrow N_{i+1}^0 + e$			
(a)	(b)	(c)	(b)
Autoionization	$N_i^{mm'} \alpha_{i,i+1}^{m'm}$	Dielectronic recombination	$n_e N_{i+1} R_{i,i+1}^{(d)mm'}$

(a): Direct Process (b): Rate Coefficient (c): Inverse Process

3.2.2 RSSOPA Model

The line structure for the bound-bound transitions becomes complicated as the number of the bound electrons increases. The lines become unresolvably close to each other. UTAOPA handles such spectrum structures based on the non-relativistic UTA method. However, for high-Z elements, the non-relativistic treatment is not adequate. The RSSOPA model uses the JJ coupling based on the relativistic UTA method.

In the RSSOPA model, the wave functions are determined by the Dirac equation:

(37) $$[c\alpha \cdot p + (\beta - I)c^2 + V(r)]\psi = \varepsilon\psi$$

where α and β are the usual Dirac matrices. V(r) is the potential. The wave function ψ has the form

(38) $$\psi_{n\kappa m} = \frac{1}{r}\begin{bmatrix} P_{n\kappa m}(r)\chi_{\kappa m} \\ iQ_{n\kappa m}(r)\chi_{-\kappa m} \end{bmatrix}$$

where P(r) and Q(r) are the radial parts and χ is a function of angular and spin coordinates in the usual notation.

The Dirac equation is solved numerically to obtain the wave functions for electron orbitals and the self-consistent potential. The photoexcitation cross sections for the configuration-to-configuration transition are calculated from the single-electron transition properties. In the configuration average approximation, the cross section can be written as

(39) $$\sigma^{bb}_{icc'}(\hbar\omega) = \frac{\pi h e^2}{mc} f_{icc'} \gamma(\hbar\omega).$$

where $f_{icc'}$ is the configuration average oscillator strength, $\gamma(\hbar\omega)$ is the line shape function. If the transition energy is assumed to be approximately the same for all lines of the transition array, the relation of the averaged array oscillator strength $f_{icc'}$ to the single-electron transition oscillator strength $f_{\alpha\beta}$ is

(40) $$f_{icc'} = q_\alpha (1 - \frac{q_\beta}{g_\beta}) f_{\alpha\beta},$$
$$f_{\alpha\beta} = \frac{2m}{\hbar\omega g_\alpha} \frac{1}{2k+1} |< \alpha \| T \| \beta >|^2,$$

where $\hbar\omega$ is the photon energy, q_α, q_β are the occupation numbers of orbital α, β respectively, k is the rank of electric multipoles, g_α is the statistical weight for initial orbital α and $< \alpha \| T \| \beta >$ is the bound-bound reduced transition matrix element.

The photoionization cross section for configuration c of ionic stage I can be written as

(41) $$\sigma_{ic} = \frac{\pi h e^2}{mc} \sum_\alpha q_\alpha \frac{df_\alpha}{d\varepsilon},$$

where the summation runs over all subshells of the configuration. $\frac{df_\alpha}{d\varepsilon}$ is the density of oscillator strength, given by

(42) $$\frac{df_\alpha}{d\varepsilon} = \frac{2m}{3\hbar\omega g_\alpha} |< \alpha \| T \| \varepsilon >|^2,$$

where $<\alpha \| T \| \varepsilon>$ is the energy-normalized transition matrix element from the initial bound orbital state α to the continuum orbital state ε.

The line shape $\gamma(\hbar\omega)$ uses the Voigt function:

$$
\begin{aligned}
&\gamma(\hbar\omega) = \frac{\sqrt{\ln 2}}{\pi\Gamma} H(a,v), \\
&H(a,v) = \frac{a}{\pi}\int_{-\infty}^{+\infty} \frac{\exp(-x^2)}{a^2 + (v-x)^2} dx, \\
&a = \sqrt{\ln 2}\,\Gamma_c / \Gamma, \\
&v = \sqrt{\ln 2}\,(\hbar\omega - E_c^{\alpha\beta}) / \Gamma, \\
&\Gamma = \Gamma_d + \Gamma_u,
\end{aligned}
\tag{43}
$$

where Γ_d is the Doppler full-width at half maximum (FWHM) given by

$$\Gamma_d = 3.858\times10^{-5}(k_B T / A)^{1/2}\hbar\omega(eV),$$

where $k_B T$ is expressed in eV, A is the atomic weight expressed in gram-moles, the transition energy is expressed in eV. Γ_c is the Lorentz FWHM due to the collisional broadening mechanism, which is calculated with the electron impact semiclassical formulas. The other two quantities $E_c^{\alpha\beta}, \Gamma_u (= 2.355\times\Delta_c^{\alpha\beta})$ are calculated in the UTA method as the following.

By the UTA method, the average energy and the standard deviation of the energy distribution of a given electronic configuration are the first and second moments of the Hamiltonian,

$$E_{av} = \left(\langle \varphi_i \mid H \mid \varphi_i \rangle\right)_{av}, \tag{44}$$

and

$$\sigma^2 = \left(\langle \varphi_i \mid H \mid \varphi_i \rangle^2 \right)_{av} - \left(\langle \varphi_i \mid H \mid \varphi_i \rangle \right)^2_{av}, \tag{45}$$

where H is the sum of the electrostatic and spin-orbit operator,

$$H = \sum_{i<j=1}^{N} \frac{e^2}{r_{ij}} + \sum_{i=1}^{N} \zeta(r_i) s_i \cdot l_i \,. \tag{46}$$

Under the j-j coupling scheme and using the Hermitian properties of the Hamiltonian, the width of the energy distribution can be expressed in the form

$$\sigma^2 = D_1 + D_2 + D_3 + D_4, \tag{47}$$

where the formula for $D_{1,2,3,4}$ are given in Reference [BOG95].

The standard deviation of the weighted line wavenumber distribution is the square root of the variance,

$$\sigma^2 = \mu_2 - \mu_1^2 \,, \tag{48}$$

where

$$\mu_n = \frac{1}{W} \sum_{ab} w_{ab} \left[\langle a \mid a \rangle - \langle b \mid b \rangle \right]^n,$$
$$W = \sum_{ab} w_{ab},$$
$$w_{ab} = \left| \left\langle a \mid \sum_i r_i \mid b \right\rangle \right|^2 .$$

The final numerical formulas for the UTA moments can be written in a concise form:

The UTA average energy

$$
\begin{aligned}
E_C^{\alpha\beta} &= E_C'^{\alpha\beta} + \delta E_C'^{\alpha\beta}, \\
E_C'^{\alpha\beta} &= D_0 + \sum_s (q_s - \delta_{s\alpha}) D_s, \\
\delta E_C'^{\alpha\beta} &= \left[\frac{q_\alpha - 1}{2 j_\alpha} - \frac{q_\beta}{2 j_\beta} \right] (F^{j_\alpha, j_\beta} + G^{j_\alpha, j_\beta}).
\end{aligned}
\tag{49}
$$

$$
\begin{aligned}
D_0 &= \left\langle j_\beta \mid h_D \mid j_\alpha \right\rangle, \\
D_s &= \left\langle j_s, j_\beta \right\rangle - \left\langle j_s, j_\alpha \right\rangle.
\end{aligned}
$$

The UTA variance

$$
(\Delta_C^{\alpha\beta})^2 = \sum_s (q_s - \delta_{s\alpha})(g_s - q_s - \delta_{s\beta}) \Delta_s^2, \tag{50}
$$

$$
\Delta_s^2 = \frac{\Delta^2 (j_s j_\alpha, j_s j_\beta)}{2 j_s - \delta_{s\alpha} - \delta_{s\beta}}, \tag{51}
$$

$$
\Delta^2 (j_s j_{\alpha,} j_s j_\beta) = A_s + B_s + C_s + D_s + E_s + F_{s\alpha\beta} + F_{s\beta\alpha,} \tag{52}
$$

where the terms $A_s, B_s, C_s, D_s, E_s, F_{s\alpha\beta}, F_{s\alpha\beta}$ are related to the radial Slater integral which are given in Reference [BOG95].

To calculate opacities, there are a number of configurations for each ion stage. For each configuration, we need to do the self-consistent field calculation, and then all the wave function calculations for both bound and free electrons, and finally we can use these wave functions to calculate the photoexcitation and photoionization cross section and the UTA width for all possible transitions. The process is very time-consuming especially for high-Z elements. To give a sense of how many calculations are involved, we use the medium-Z (Z=50) as an example. Each ion stage has an average of 20

configurations (at least), and each configuration has an average of 4 orbitals, and for each orbital, there are an average 15 transitions (5 for the bound-bound transition (at least) and 10 for the bound-free transition). Therefore, there are a total of 60,000 transitions to be calculated, plus the iterations needed for solving the Dirac equation. Significant computer time is needed for these calculations.

Nevertheless, this model has a nice feature under the average configuration approximation, that is, the calculation for each configuration is independent. Therefore, the parallel computing technique can come into play. In this thesis, we implement the RSSOPA model using MPI. The flow diagram of RSSOPA model is shown in Fig. 3.2.1. Time measurements for parallel computations are given in Section 4.2.

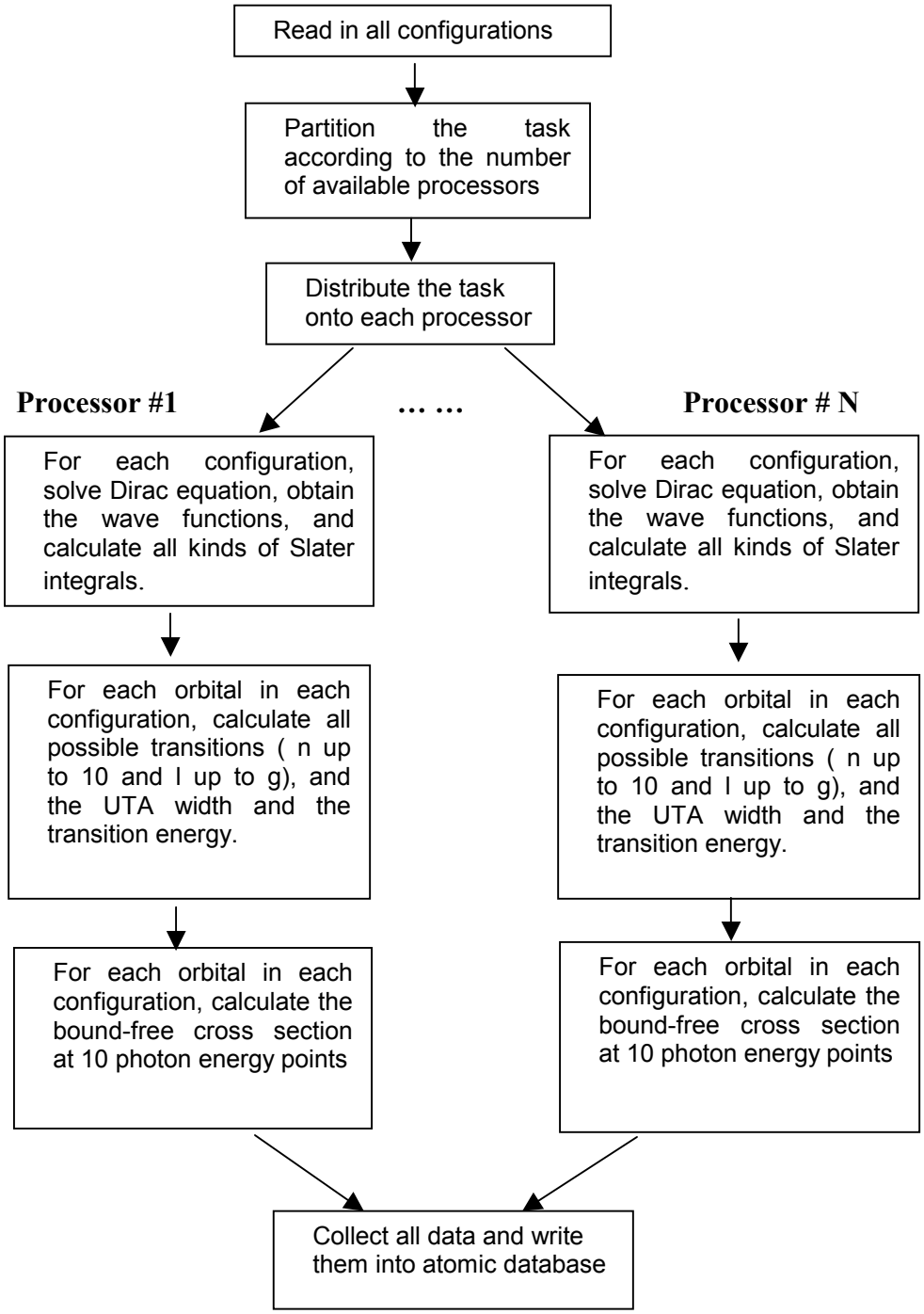


Fig. 3.2.1 Flow diagram of RSSOPA parallel coding

Chapter 4

Implementation of Distributed Atomic Data Computing System

Advances in architectures, software and networks have shifted the traditional computing environment to the distributed environment. The advent of high-speed networks and the needs of harnessing more computer resources in high performance scientific and engineering applications has led to the possibility of federating resources such as compute power, data storage and networks into computational grids [GRD99]. Computational grid software infrastructures such as Globus [FK97], Legion [LG96], and Condor [LLM88] provide abstractions to give users the ability to run applications on a heterogeneous set of machines as they once did on a single high performance platform. These infrastructures also provide services such as security, communication, managing distributed applications and remote data transfer.

While metacomputing systems such as Globus, Legion and Condor provide low level grid APIs that can be used to implement low tier services, the computational grid

can also build on commodity network technologies, such as CORBA, COM and Java Beans. These technologies are being used to construct multi-tiered architectures. The top tier of this model always consists of components such as graphic user interfaces. The middle tier consists of program logic and other high-level services such as load balancing and integration of legacy systems. The middle tier mediates between sophisticated back-end data services and simple front ends. The bottom tier provides data services from relational and object databases. The decomposition of application functionality into separate presentation, application and data service results in a distributed computing architecture for computational grids.

In this chapter, we describe implementation of the distributed atomic data computing system based on the commodity architecture (as shown in Fig. 2.4.2). The layout of this chapter is as follows:

In Section 4.1, we first give some results on the atomic data computing. After verifying the accuracy of the ATBASE codes by comparing with other theoretical results and experiments, the importance of JJ coupling for medium- and high-Z elements are shown in the atomic data calculation and the spectrum analysis. In Section 4.2, we give a description of the parallel computing of the atomic data under the RSSUTA model. Through several experimental calculations, we find that the speed-up factor increases almost linearly with the number of processors. This characteristic results from the minimization of the requirements of communications. In Section 4.3, we discuss the atomic database design and prove that the object data model is more suitable than the

relational data model to simulate the atomic data set. A sample of SQL script to create the object table schema is presented. The coupling of the three-tier of our data computing system is based on the use of CORBA middleware (for the web-based project MIXOPA, we use Enterprise Java Beans). We also give a description of the interfaces used in the CORBA framework. In the following sections, we discuss the graphic user interfaces for the four application components in this project, which are: the atomic data calculation, the EOS and opacity calculation, the data visualization tool and the spectrum analysis. In Section 4.4, we show the graphic user interfaces for the first three components. For the spectrum analysis, we show the graphic user interface in Section 4.5. Implementation of each component is emphasized. Detailed usage information is given in the User's Manual. Finally, we give a very brief description of web-based projects in Section 4.6.

4.1 Results on Atomic Data Computing

In this section, we first compare the atomic data calculated by ATBASE with other theoretical results and experiments to test the accuracy of the ATBASE codes, then we show the importance of jj coupling with increasing of the nuclear charge Z in the atomic data calculations and the spectrum and opacity calculations.

4.1.1 Verification of the atomic data generated by ATBASE

We list some numerical data of energy levels calculated by ATBASE using the DTA model and the Cowan's code from low-Z element (Carbon Z=6) to high-Z element (Gold Z=79) in Table 4.1.1. As we can see, the agreement between ATBASE and Cowan's code are very good. The difference of energy levels calculated by ATBASE and the Cowan's code is within 2% and the difference of oscillator strengths is within 10%.

For the spectrum analysis, we care about the accuracy of the detailed transition lines. However, for the opacity calculation used by hydrodynamic simulations, such detailed line structures are not necessary because they may be washed out after integration with the radiation field. The UTA method is a good approximation for this purpose. In Table 4.1.2, we calculate the UTA results of transition energy for Ni-, Cu-, Zn- like W ions. We can see the ATBASE results agree with the STA results and experiments. Because ATBASE does not give the transition width explicitly, we can not compare the width with the STA results and experiment.

In order to test the accuracy of the transition width, we compare the results calculated by the fully relativistic RSSUTA model with Bauche's results [BBK85] in Table 4.1.3. We chose the $3d^9 - 3d^8 4p$ transition array in the spectrum of Co-like sequence of W (Z=74). From Table 4.1.3, we can see the comparison of both wave position and width are satisfactory.

Table 4.1.1 Detailed energy levels comparison between ATBASE code and Cowan's code.

$$\mathbf{C}^{3+}\ 1s^2 2s^2 - 1s^2 2s^1 2p^1$$

Upper J	Lower J'	E(Atbase)	E(Cowan)	f(Atbase)	f(Cowan)
2S}2S 0.5	=> 1S}2P 0.5	2136.7278	2124.8903	9.084E-03	9.100E-3
2S}2S 0.5	=> 1S}2P 1.5	2132.9080	2121.0379	1.894E-02	1.900E-2
2S}2S 0.5	=> 3S}2P 0.5	1253.2146	1237.2361	3.621E-01	3.677E-1
2S}2S 0.5	=> 3S}2P 1.5	1252.3782	1236.4062	7.233E-01	7.347E-1

$$\mathbf{Ar}^{9+}\ 1s^2 2s^2 2p^6 - 1s^2 2s^2 2p^5 3d$$

1S}1S 0.0	=> 2P}3P 1.0	42.5717	42.5649	4.839E-03	5.100E-3
1S}1S 0.0	=> 2P}3D 1.0	42.0823	42.0864	1.196E-01	1.300E-1
1S}1S 0.0	=> 2P}1P 1.0	41.4601	41.4831	2.318E+00	2.279E+00

$$\mathbf{Ge}^{27+}\ 1s^2 2s^2 2p^6 3s^2 3p^6 3d^9 - 1s^2 2s^2 2p^6 3s^2 3p^6 3d^8 4f$$

2D}2D 1.5	=> 3F}4F 2.5	149.1955	149.3197	1.196E-02	1.640E-02
2D}2D 1.5	=> 3F}2P 0.5	148.0378	148.0497	8.886E-02	8.680E-02
2D}2D 2.5	=> 3F}4G 2.5	147.0757	147.0887	7.172E-03	7.100E-03
2D}2D 2.5	=> 3F}2F 2.5	146.6788	146.7086	2.186E-01	2.079E-01
2D}2D 1.5	=> 1D}2P 1.5	144.1430	144.1137	1.868E-02	1.840E-02
2D}2D 2.5	=> 1D}2D 2.5	143.2658	143.241	1.901E-01	1.873E-01
2D}2D 2.5	=> 3P}2F 3.5	142.1415	142.1176	5.679E-01	5.612E-01

$$\mathbf{Xe}^{26+}\ 1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} - 1s^2 2s^2 2p^6 3s^2 3p^6 3d^9 4f$$

1S}1S 0.0	=> 2D}3P 1.0	14.7751	14.7651	9.769E-03	0.0099
1S}1S 0.0	=> 2D}3D 1.0	14.6135	14.6124	4.352E-01	0.5270
1S}1S 0.0	=> 2D}1P 1.0	14.1986	14.2306	6.322E+00	6.2098

$$\mathbf{Au}^{51+}\ 1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} - 1s^2 2s^2 2p^6 3s^2 3p^6 3d^9 4f$$

1S}1S 0.0	=> 2D}3P 1.0	5.0066	5.0028	6.496E-04	2.000E-04
1S}1S 0.0	=> 2D}3D 1.0	4.9319	4.9324	2.288E+00	2.498E+00
1S}1S 0.0	=> 2D}1P 1.0	4.7572	4.7619	6.019E+00	5.795E+00

Table 4.1.2 Comparison of UTA calculated transition energies for W(Z=74)

Ni-like

Transition	E(ATBASE)	E(STA)	E(Exp)
3d – 5f	4.3737	4.046	4.405
3d – 6f	3.8500	3.8792	3.878
3d – 7f	3.5921	3.6199	3.620

Cu-like

Transition	E(ATBASE)	E(STA)	E(Exp)
3d – 5f	4.4226	4.455	4.456
3d – 6f	3.9032	3.935	3.932
3d – 7f	3.6468	3.677	3.676

Zn-like

Transition	E(ATBASE)	E(STA)	E(Exp)
3d – 5f	4.472	4.508	4.506
3d – 6f	3.9573	3.993	3.990
3d – 7f	3.7028	3.737	--

Table 4.1.3 Comparison of details of the position and widths of the subarray of $3d^9 - 3d^8 4p$ transition for W(Z=74) between RSSUTA model and Bauche's results. (a) represents the Bauche's results [BBK85]; (b) represents the RSSUTA reults.

Subarray	*One-electron transition*	*Wave number(Å)*		*FWHM(Å)*	
		(a)	*(b)*	*(a)*	*(b)*
$d^3_{3/2}d^6_{5/2} - d^2_{3/2}d^6_{5/2}p_{1/2}$	$3d_{3/2} - 4p_{1/2}$	6.9666	6.9677	0.075	0.083
$d^3_{3/2}d^6_{5/2} - d^2_{3/2}d^6_{5/2}p_{3/2}$	$3d_{3/2} - 4p_{3/2}$	6.5838	6.5642	0.068	0.064
$d^3_{3/2}d^6_{5/2} - d^3_{3/2}d^5_{5/2}p_{3/2}$	$3d_{5/2} - 4p_{3/2}$	6.8155	6.8211	0.046	0.058
$d^4_{3/2}d^5_{5/2} - d^3_{3/2}d^5_{5/2}p_{1/2}$	$3d_{3/2} - 4p_{1/2}$	6.9665	6.9715	0.058	0.047
$d^4_{3/2}d^5_{5/2} - d^3_{3/2}d^5_{5/2}p_{3/2}$	$3d_{3/2} - 4p_{3/2}$	6.5835	6.5679	0.053	0.038
$d^4_{3/2}d^5_{5/2} - d^4_{3/2}d^4_{5/2}p_{3/2}$	$3d_{5/2} - 4p_{3/2}$	6.8227	6.8249	0.057	0.042

4.1.2 Evolution of the transition pattern from low Z to high Z

Under the condition that the electrostatic interaction between electrons are much stronger than the interaction between the spin of an electron and its own orbital motion, the appropriate coupling scheme is LS coupling. With increasing nuclear charge Z, the spin-orbit interactions become increasingly more important. When these interactions become much stronger than the Coulomb terms, the coupling conditions approach JJ coupling. In the JJ coupling, basis functions are formed by first coupling the spin of each electron to its own orbital angular momentum, and then coupling together the various resultant j in an arbitrary order to obtain the total angular momentum J.

We use the simple transition array $3d^8 4s - 3d^8 4p$ along the isoelectronic sequence for Kr (Z=36), Mo (Z=42) and Pr (Z=59) to demonstrate the importance of JJ coupling. We show that the spectrum is split with increasing nuclear charge Z. This feature can be derived using the crude assumption that the external 3d, 4s, and 4p orbitals are hydrogenic with the effective nuclear charge $Z^* \cong Z - 26$. Under this assumption, the Slater integrals of the electron pair (3d, 3d), (3d, 4s) and (3d, 4p) are proportional to Z^*. However, the spin orbit integrals ζ_{3d}, ζ_{4p} for orbitals 3d and 4p are proportional to $(Z^*)^4$. Therefore, it is clear that the spin-orbit integrals become predominant when nuclear charge Z increases and the transition array $3d^8 4s - 3d^8 4p$ splits into two subarrays, that is, the longer wavelength transition $4s_{1/2} - 4p_{1/2}$ and the shorter wavelength transition $4s_{1/2} - 4p_{3/2}$. Two atomic models are used. For the DTA model, we use the Cowan's code for computing the wavelengths and the transition strengths with relativistic corrections. For the UTA model, we use the RSSUTA model to obtain the UTA transition position and width. Fig. 4.1.1(a)-(c) show the evolution of the pattern. The RSSUTA numerical data used to calculate the Gaussian profile is listed in Table 4.1.4.

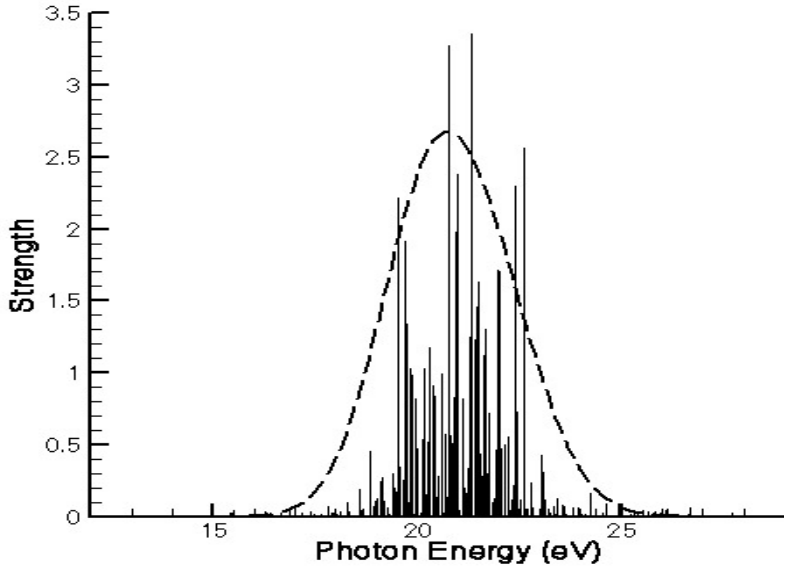


Fig. 4.1.1(a) Spectrum for Kr $3d^8 4s - 3d^8 4p$

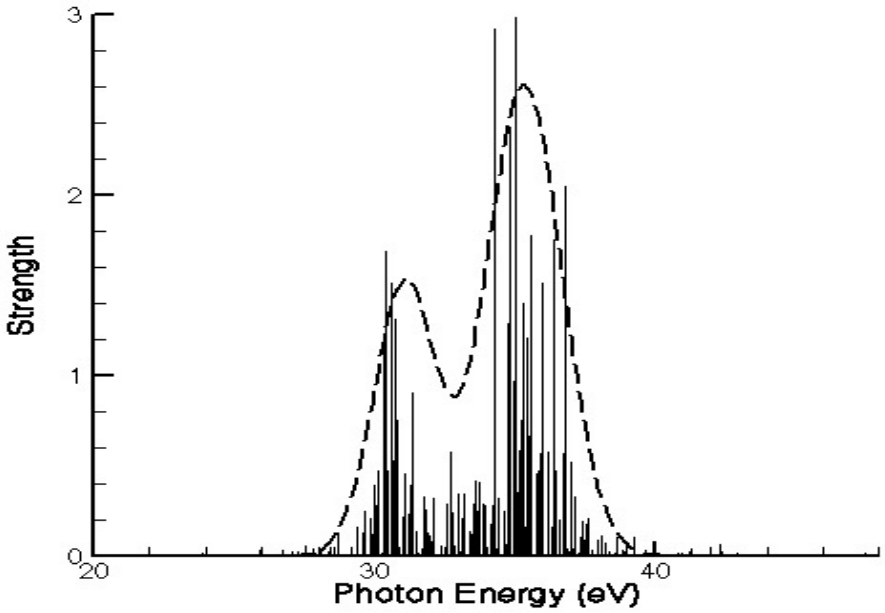


Fig. 4.1.1(b) Spectrum for Mo $3d^8 4s - 3d^8 4p$

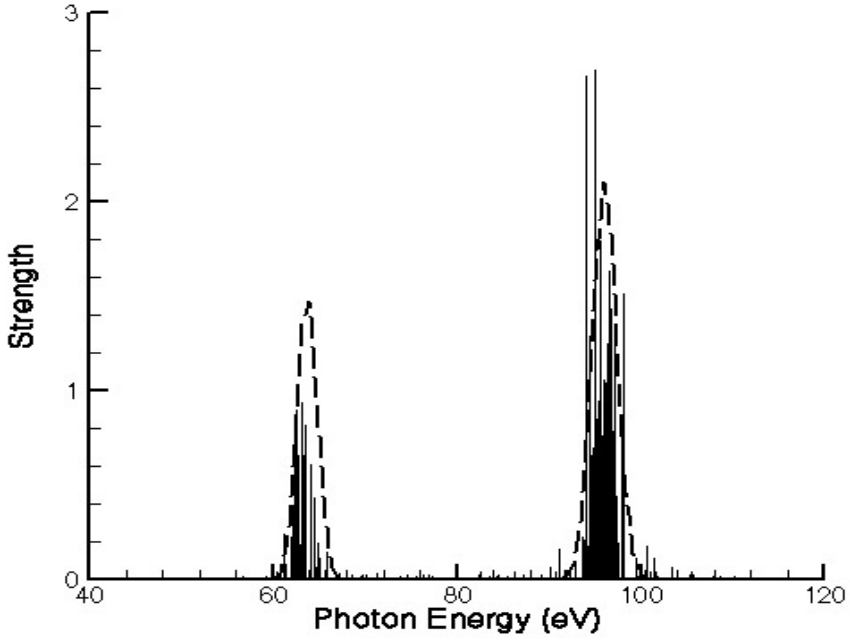


Fig. 4.1.1(c) Spectrum for Pr $3d^8 4s - 3d^8 4p$

Fig. 4.1.1 Calculated spectra in the $3d^8 4s - 3d^8 4p$ series for Kr, Mo, Pr. The vertical lines are calculated using the Cowan's code. The dashed lines are calculated using the RSSUTA model. The numerical data used for the Gaussian shapes are listed in Table 4.1.4. These figures clearly show evolution of the spectrum pattern with increasing nuclear charge Z. The transition array is split into two subarrays, that is, $4s_{1/2} - 4p_{1/2}$ and $4s_{1/2} - 4p_{3/2}$.

	Kr (Z=36) $3d^8 4s - 3d^8 4p$		
	Transition Energy (eV)	Variance	Oscillate Strength
(a)	20.08 *	6.68	3.417411E-04
	21.26 **	14.66	4.563104E-04
(b)	19.87	6.68	3.510421E-04
	21.36	14.81	6.915320E-04
(c)	19.66	6.68	1.515523E-04
	21.46	14.79	5.588995E-04
	Mo (Z=42) $3d^8 4s - 3d^8 4p$		
	Transition Energy (eV)	Variance	Oscillate Strength
(a)	31.16	5.54	6.933621E-04
	35.19	13.06	9.395431E-04
(b)	31.05	5.54	6.910049E-04
	35.36	13.05	1.430757E-03
(c)	31.02	5.54	2.735523E-04
	35.47	13.06	1.161781E-03
	Pr (Z=59) $3d^8 4s - 3d^8 4p$		
	Transition Energy (eV)	Variance	Oscillate Strength
(a)	64.24	3.7	2.308677E-03
	95.83	11.9	3.088790E-03
(b)	63.72	3.8	2.249766E-03
	96.08	11.8	4.729155E-03
(c)	63.21	3.7	2.292728E-03
	96.32	11.8	3.860142E-03

(a) : $3d_{3/2}^2 3d_{5/2}^6 4s_{1/2}$ (b): $3d_{3/2}^3 3d_{5/2}^5 4s_{1/2}$ (c): $3d_{3/2}^4 3d_{5/2}^4 4s_{1/2}$

(*): $4s_{1/2} - 4p_{1/2}$ (**): $4s_{1/2} - 4p_{3/2}$

Table 4.1.4 Numerical transition data for array $3d^8 4s - 3d^8 4p$ for Kr, Mo, Pr

4.1.3 The effect of JJ coupling on spectrum simulations

In Fig. 4.1.2, we compare the ATBASE DTA transmission calculation with the UTA result for an Al plasma. As we expect, there are no split subarrays. The different peaks are corresponding to different ion stages (B-like, C-like, N-like ions). Comparing with experiment, we know that the DTA results for both transition energy position and the strength agree with experiment very well. For UTA simulations, we notice that: 1) under the same plasma condition (T=40eV), the UTA spectrum shift toward higher energy side. This means the ATBASE UTA predicts more over-stripped ions. On the contrary, we can see the ATBASE UTA simulation under the plasma condition T=30eV underestimates the ionization degree, and therefore the spectrum is toward the low energy side. The line strength of ATBASE UTA simulation under T=35eV is closer to experiment. 2) the energy transition positions are different, comparing with the ATBASE DTA calculation and experiment. The transition energy calculated by ATBASE UTA method is not accurate enough to form a Gaussian profile to encapsulate those detailed transition lines. We can also see this from other comparisons.

In Fig. 4.1.3, we compare the transmission spectrum calculated by several theoretical models and experiment for a Nb plasma (T=47eV, D=0.026 g/cm^3). We can see that the RSSOPA and STA results agree with experiment much better than the ATBASE UTA. Actually, the ATBASE UTA has no such detailed structures. The reason is that the ATBASE UTA uses the non-relativistic LS coupling UTA method for

the transition array. For example, for transition 3d-2p, there are three lines under JJ coupling :

$$3d^{5/2} - 2p^{3/2}, 3d^{3/2} - 2p^{1/2}, 3d^{3/2} - 2p^{3/2}$$

However, under LS coupling, there is only one transition array 3d-2p. That is why we can only see big bundles around 2200eV and 2450eV. More detailed comparison between theoretical simulations and experiment is given in Section 4.5 when we discuss the implementation of the spectrum analysis module. From this example, we can see clearly the effect of JJ coupling on the medium to high Z spectrum.

In Fig. 4.1.4, we do the similar comparison for Ge plasma (T=76eV D=0.054 g/cm^3). Again, we can see the ATBASE UTA method produces less accurate transition energy positions and has no such detailed spectral resolution. This experimental data is also used to test the implementation of our spectrum analysis module in Section 4.5.

We show a comparison of an opacity calculation for a Au plasma in Fig. 4.1.5. Using this example we argue that enough configurations are needed to the opacity calculations. Three theoretical models are used: STA by A Bar-Shalom, EOSOPA by P.Wang and the Average Atom (AA) model of this work. The numerical solution of the AA model is similar to the RSSOPA model except that the potential is given by an empirical form using Density Function Theory (DFT). In the AA model, only one virtual average atom exists in the plasma. We can see some interesting differences. The ATBASE UTA opacities are below the STA results, which is instead lower than AA

results. The reason is that the current ATBASE UTA does not include enough configurations for low ionized ions, which contribute mainly to the low energy part of the opacity. On the other hand , the AA model overestimates the opacity because the AA model assumes all the ions have equal ionization energy, and what is more, the AA model has no detailed structure. However, they all have similar continuum opacity above 1000eV.

In a summary of the section, we can make a conclusion that the ATBASE code does very good calculations from low-Z to medium-Z elements under the Detailed Term Accounting (DTA) model, and for the opacity table generation, the ATBASE UTA model can also provide very reasonable results. However, for the spectrum analysis characterized by unresolved overlap transition structures, the ATBASE UTA model is less accurate than the RSSUTA model. The RSSUTA model is more appropriate for high Z elements because of its fully relativistic treatment. The summary of numeric codes for atomic data calculations and their capabilities are given in Appendix C.

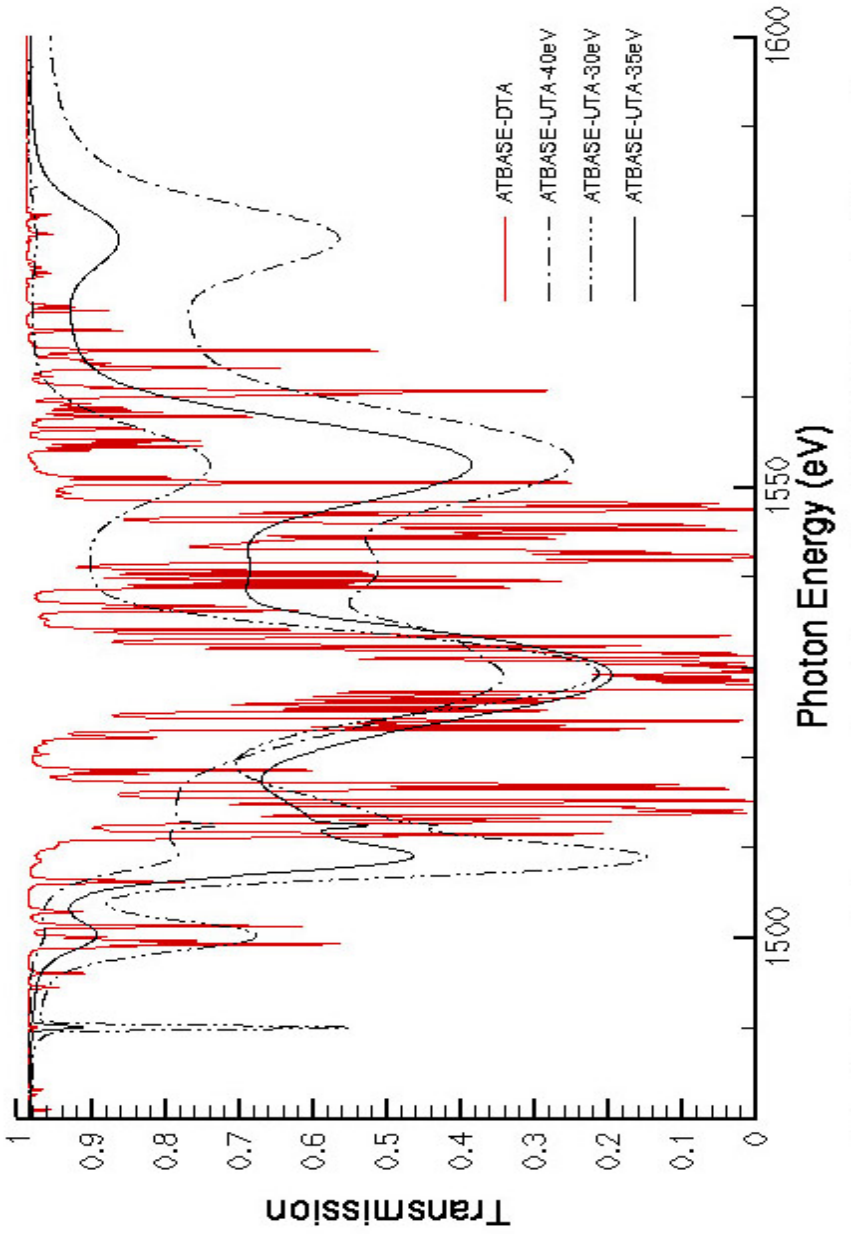


Fig. 4.1.2 Transmission comparison between the ATBASE DTA calculation and UTA calculation for Al plasma (T=40eV, D=0.013 g/cm^3).

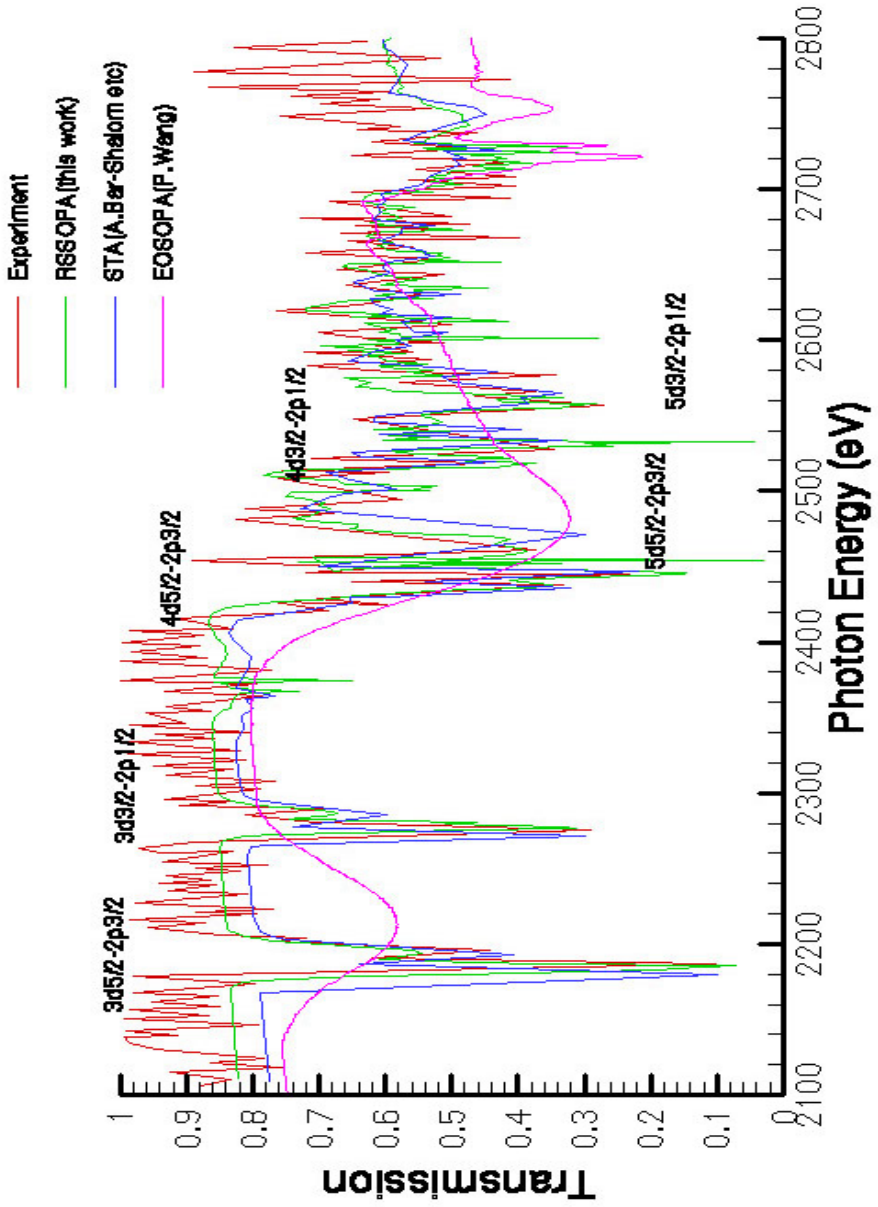


Fig. 4.1.3 Comparison of the absorption data calculated by several theoretical models and experiment for the Nb spectrum. The plasma condition is T=47eV, D=0.026 g/cm^3.

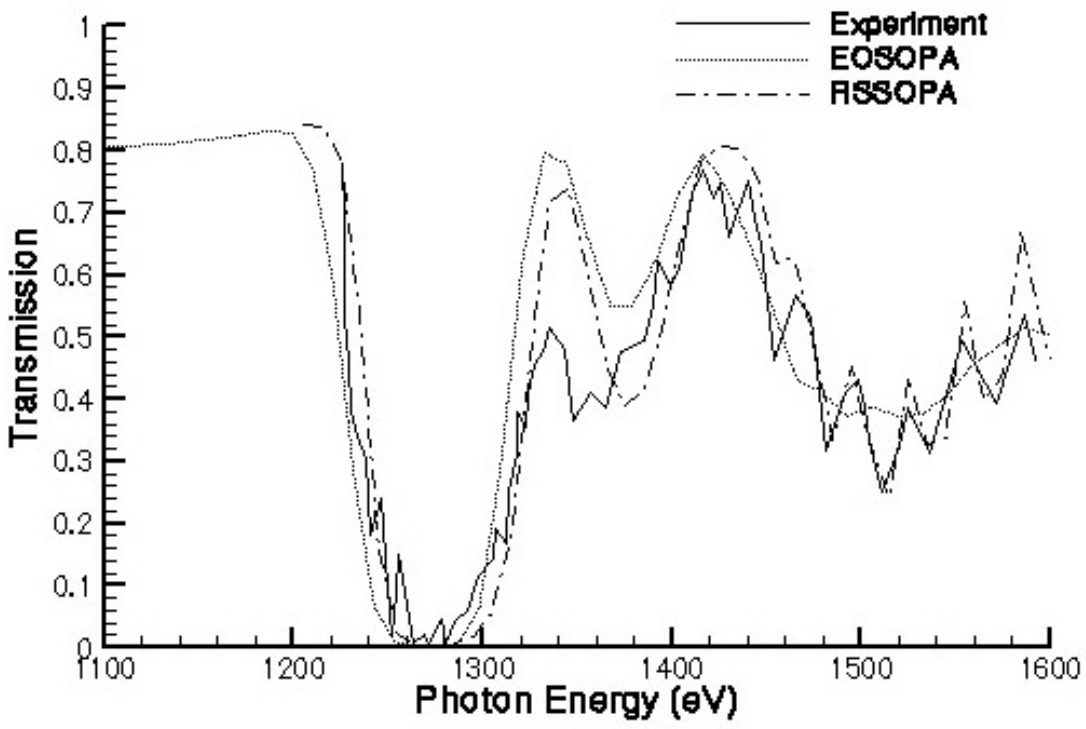


Fig. 4.1.4 Comparison of transmission calculations by ATBASE UTA and RSSOPA with experiment for Ge (T=76eV, D=0.054 g/cm^3).

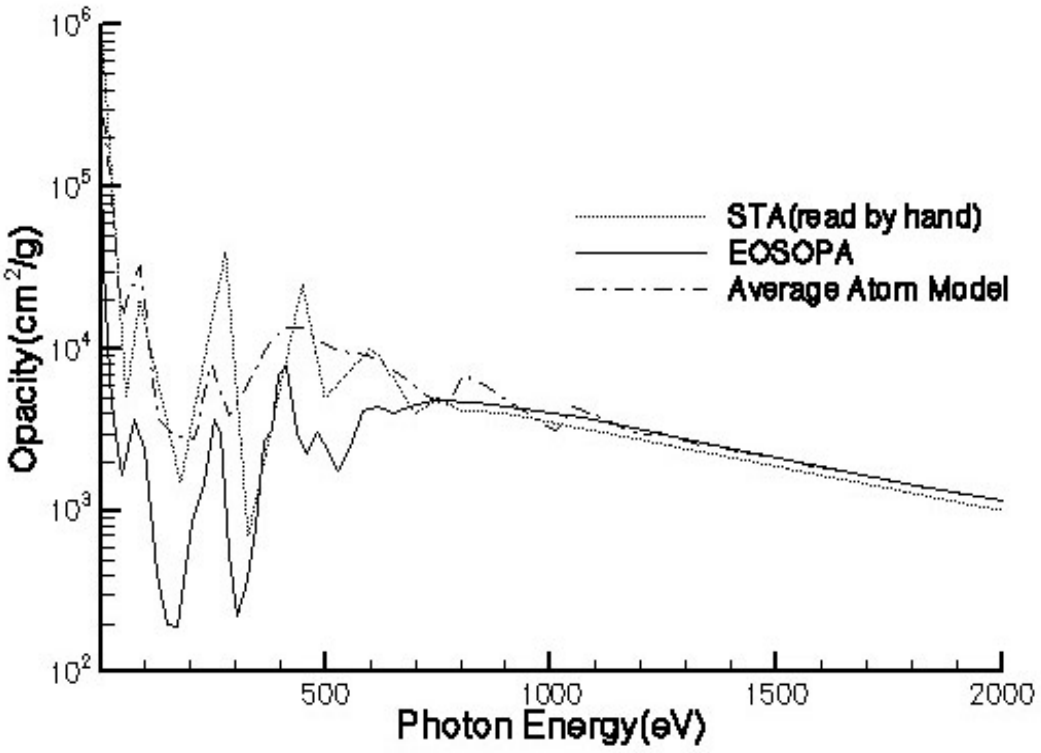


Fig. 4.1.5 Opacity comparison for Au plasma (T=100eV, D=0.1 g/cm^3). STA data are read by hand

4.2 Parallel Computing

As shown in Section 3.1, Schrödinger's equation for atoms is a kinetic equation for a many-body system. As the many-body problem is solved to a high level of accuracy, the prediction of atomic properties is a challenging interaction between computational techniques and theoretical physics. Large demands on computer resources are required to obtain accurate atomic data. Using nonrelativistic with low-order relativistic corrections (MCHF) or a fully relativistic Driac-Fock theory (MCDF), a sparse interaction matrix as large as 100,000×100,000 is needed to solve for eignvalues and eignfunctions. It is an astonishing job for a single processor. Since the computation of each block of the sparse matrix is independent from any other, it is possible to distribute the computation of each block across processors. C. Froese Fischer has performed a lot of experiments on the accurate atomic data calculation from implementation on parallel-vector computer (Cray) to a cluster of IBM/RS6000 using PVM and finally to MPI on IBM SP2 [FTB94].

While C. Froese Fischer emphasizes high performance computing for accurate atomic data for diagnosis, we emphasize the atomic data for the opacity calculation and high Z elements. To calculate opacities, there are a lot of electron configurations needed to be considered. If we ignore the interactions between configurations, the calculation for each configuration is independent under the RSSOPA model, as discussed in Section

3.2.2. Therefore, the communication cost can be minimized and hence the throughput can be maximized. The code structure is listed in Table 4.2.1.

Table 4.2.1 The pseudo-code structure for atomic data parallel computing

```
Do i = 0, ionz-1    // loop over the ion stages
   Primary node reads all relativistic configurations and
   partition configurations according to available processors
   each processor has a pair of configuration index
   ip0: // low boundary        ip1: // high boundary
   do j = ip0, ip1  // loop over the configurations
        subtask for UTA calculation for each configuration
   end do
   collect the output from each processor
end do
```

We performed several runs to measure performance of the RSSUTA parallel code implemented in MPI. We use Ti-like (22 electrons) W (Z = 74) as a sample. For this ion stage, there are 262 relativistic electron configurations. The same calculations have been repeated on1, 4, 8, 16, 32, 64, 128, 256 processors using the NPACI IBM SP machine. Wall clock time has been recorded for each processor. The calculation time is the average of all wall clock time for each run, which is listed in Table 4.2.2. Fig. 4.2.1 shows the speed-up versus the number of processors. From the figure, we can see that the speed-up is almost linearly increasing with the number of processors.

Table 4.2.2 Calculation time and speed-up versus the number of processors

Number of processors	Calculation time in seconds	Speed -up
1	17695.82	1
4	4414.94	4.008
8	2245.24	7.881
16	1098.95	16.102
32	551.79	32.07
64	277.75	63.711
128	143.14	123.626
256	74.29	238.2

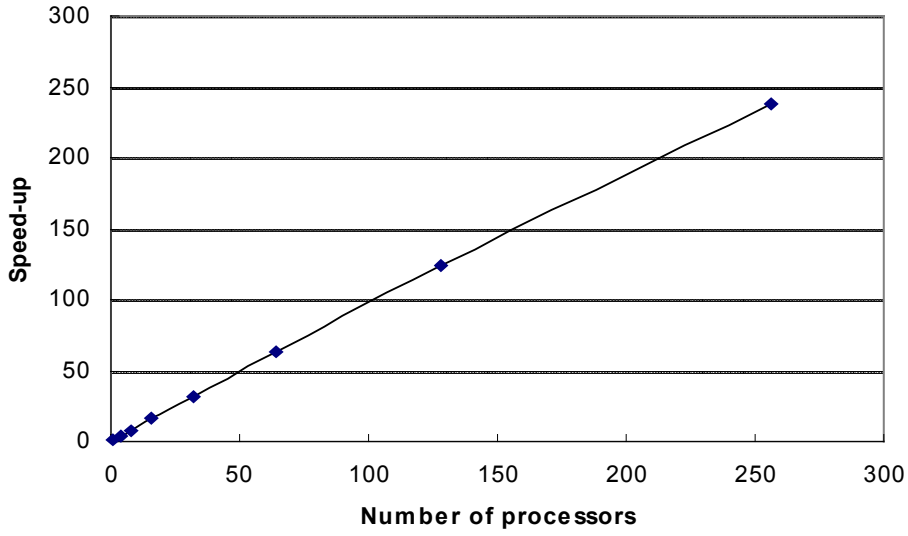


Fig. 4.2.1 The speed-up versus the number of processors.

4.3 Database Design and Distributed Computing

In a traditional computing environment, the computing resource is tightly coupled with local file systems, i.e. using local disks as data storage, such as in the spectrum and opacity calculations. In these calculations [PW96][MCW98], the conventional approach is to generate the atomic data in a flat file first, and then read the whole file into the opacity or spectrum applications. This approach requires large memory to load the whole data set at once but the execution is fast. The major drawback of this approach is that it is difficult to track down the detailed information that is involved in the calculation. It has to record the detailed information when the program is running and to use a post-processor to reveal them. It is difficult to answer questions such as how many configurations have been involved in the calculation and what are the transition lines appearing in the spectrum. Another potential difficulty in the atomic data calculation by using flat files on local file systems is the atomic data size may exceed the disk capacity or run out of the memory capacity. Depending on the atomic model, the application can generate huge amounts of data containing information about the atomic structure and the rate coefficients, especially for high Z elements. This could be a problem when the spectrum and opacity application try to load these huge amounts of data. They can exhaust the memory resource easily even though only a small part of the data is actually used. To handle this problem, ATBASE has to propose an approximation method by averaging the detailed spectrum structures to reduce the data size in order that the atomic data can fit into the available memory

capacity [PW96]. To overcome these problems, we use a data management system for storing our atomic data.

There are three types of DBMS that are widely used: relational database management system, object-oriented database management system and relational-object database management system. Object-oriented databases (OODBs) have a great advantage for scientific data management. Usually, a scientific database can contain large volumes of scientific data involving disparate types such as numeric data, image data, spatial data and temporal data. The conventional record-oriented database systems lack the support for the data models and data manipulations that match scientific data structures and operations. OODBs can directly support complex objects for capturing hierarchical structures and OODBs generally have collection types, such as lists and arrays, that are a better basis for the dimensional data common in scientific applications. With the OODBs's inherent extensibility, object-oriented databases (OODBs) are in many ways a better match for scientific data management than conventional record-oriented database systems.

In this project, we use Oracle object-relational database management system (DBMS) for quick development. The advantage of using the commercial database management systems is that they are well established and support most of network computing technologies, such as CORBA and Java Beans. The Oracle inherent network computing architecture makes our implementation much easier.

4.3.1 Database Design

We focus on the atomic data required by ICF applications, that is, the opacity data and spectrum data. We have built 23 tables in the database thus far. For the mixed opacity calculation, we store the opacity data for elements from Hydrogen (Z=1) to Calcium (Z=20). The opacity data contains information such as the average ionization stage, electron and ion pressures, Rossland opacity, Planck emission and absorption opacities and so on. The mixed opacities can be calculated from these basis tables using the linear interpolation method, as described in Section 4.6. This part of the database supports web-based projects MIXOPA and OPAVIEWER. For opacity and spectrum calculations, we store the transition energy, transition oscillate strength, UTA width, configuration average energy and other information in the relation-object tables supported by the Oracle DBMS. These atomic data stored in the database enable us fairly easily to track down all detailed information in the spectrum analysis using appropriate queries, as shown in examples of Section 4.4.

The Structured Query Language (SQL) script used to generate some essential tables are listed as the following:

Table 4.3.1 The SQL script file used to generate the database schema

```
Create or replace type RELORB_OBJTYP as object
(n number(2),  k number(2)) /
create or replace type PHTNCS_OBJTYP as object
( phtne    number,  phtnos   number) /
create table RELCFGAVGENG
(OID number(5) CONSTRAINT relcfgavgeng_oid_fk REFERENCES
OPERATION(OID),
RelConfig   varchar2(160) CONSTRAINT relcfgavgeng_relconfig_nn NOT NULL,
Ionstage    number(3) CONSTRAINT relcfgavgeng_ionstage_nn NOT NULL,
Cfgweight   number    CONSTRAINT relcfgavgeng_cfgweight_nn NOT NULL,
Avgenergy   number    CONSTRAINT relcfgavgeng_avgenergy_nn  NOT NULL,
PRIMARY KEY(OID, RelConfig) ) /
Create or replace type PHTNCS_NTABTYP as TABLE of PHTNCS_OBJTYP /
Create or replace type RELORBPHTN_OBJTYP as object
(relorb  RELORB_OBJTYP,
occup   number, orbeng  number, qfect   number, radi    number,
phtncs_ntab  PHTNCS_NTABTYP) /
create or replace type RELORBPHTN_REF_NTABTYP as table of REF
RELORBPHTN_OBJTYP /
create or replace type RELCFGPHTNION_OBJTYP as object
(OID number(5),  RelConfig   varchar2(160),
relorbphtncs_ntab  RELORBPHTN_REF_NTABTYP) /
create table RELCFGPHTNION of RELCFGPHTNION_OBJTYP
```

```
(OID CONSTRAINT relcfgphtnion_oid_fk REFERENCES OPERATION(OID),
PRIMARY KEY (OID,RelConfig) )
Nested table relorbphtncs_ntab store as relorbphtncs_store_ntab /
Create or replace type RELTRAN_OBJTYP as object
( relorbi  RELORB_OBJTYP,  relorbj  RELORB_OBJTYP,
traneng  number,  tranos  number,  utawid  number) /
create or replace type RELTRAN_NTABTYP as table of RELTRAN_OBJTYP /
create table RELCFGPHTNEXCT
(OID number(5) CONSTRAINT relcfgphtnexct_oid_fk REFERENCES
OPERATION(OID),
RelConfig  varchar2(160) CONSTRAINT relcfgphtnexct_relconfig_nn NOT NULL,
Reltran_ntab  RELTRAN_NTABTYP)
Nested table reltran_ntab  store as reltran_store_ntab /
```

As shown in the above table, the most important three data tables are RELCFGAVGENG, RELCFGPHTNION and RELCFGPHTNEXCT. The RELCFGAVGENG table stores the relativistic configuration average energies for all ion stages. Under the local thermal equilibrium (LTE) approximation, only this table is needed to compute the state populations. These configuration energy data are transferred into the spectrum analysis module to solve the Saha equation to determine the configuration populations, and then the corresponding photoexcitation data from the RELCFGPHTNEXCT table and the corresponding photoionization data from the RELCFGPHTNION table are extracted from the database. Because the size of RELCFGAVGENG table is small and the program only loads those important

photoexcitation and photoionization data, this approach speeds up the data transfer and saves the memory usage. The database access diagram for the opacity and spectrum analysis is shown in Fig. 4.3.1.

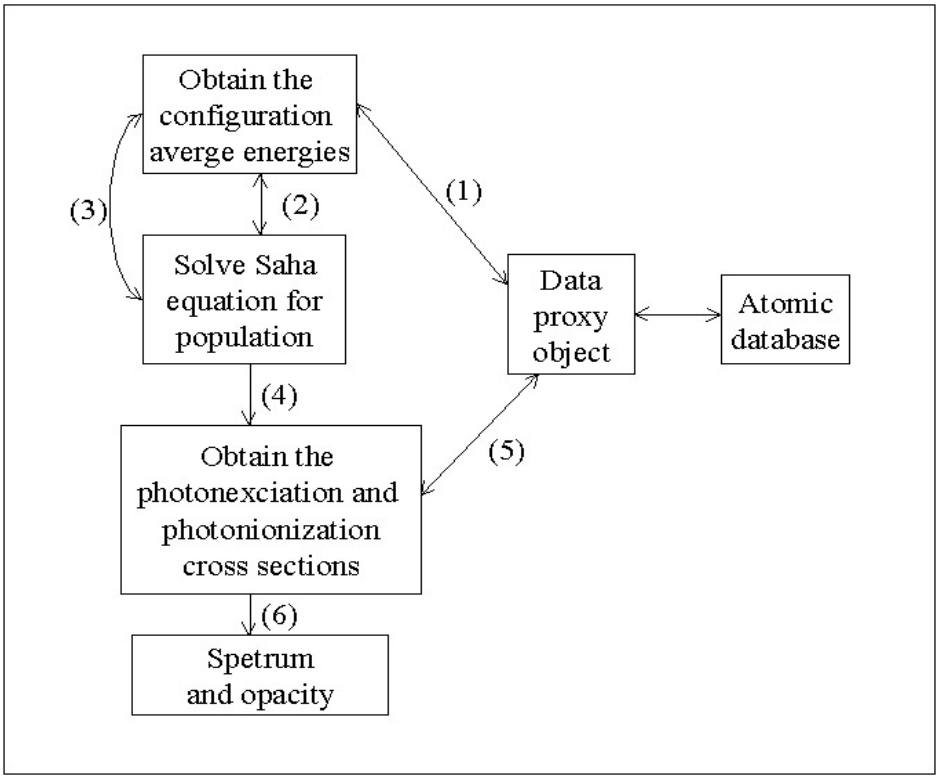


Fig. 4.3.1 The database access diagram for the opacity and spectrum calculation

The application module first establishes connection to the atomic database. After the user specifies the plasma condition, the module retrieves the 10 lowest configuration average energies for each ion stage to guess the important ion stages that may exist in this plasma condition (Step 1), we then solve the Saha equation using the minimum set of configuration energy to obtain the significant ion stages which have populations greater than some criteria (for example, crit = 0.001) (Step 2). The module

retrieves all configurations for these significant ion stages (Step 3) and repeats the process of solving the Saha equation to obtain more detailed populations. The important configurations are then determined by some criteria (Step 4). For these selected important configurations, the module contacts the data proxy object to get the photoexcitation and photoionization cross sections (Step 5). Using the population and the cross section, the module finally constructs the spectrum and opacity (Step 6).

Because we can use the SQL query language against the database, we can easily identify the spectrum structure and other detailed information. In Section 4.5, we give several examples. More details can be found in the User's Manual.

4.3.2 Distributed object computing

The database serves as a data provider in our three-tier architecture. No significant application logic is done in the database server. An agent object that is responsible for all kinds of data requests is deployed in the Oracle DBMS under the CORBA framework. The application server obtains the required atomic data through the data agent object and does most of the application logic, such as solving the LTE Saha equation and computing the opacity. CORBA introduces an Interface Definition Language (IDL) for describing the interface of distributed objects. After the interfaces have been determined, we can use the IDL compiler to generate the client-side stub and the server-side skeleton, which are responsible for the network communication and data serialization. Detailed information about CORBA architecture has been discussed in

Section 2.2.1. The interfaces used for client and server communication in Project JEOSOPA are listed in the following table:

Table 4.3.2 The CORBA interfaces used for the atomic data transfer

```
public interface SPACORBAInterface extends org.omg.CORBA.Object {
   String getAllElements();
   Vector getAvgEng(String tableName, int ionstage);
   Vector getPreAvgEng(String tableName, int ionstage);
   Vector getIoniEng(String element);
   Vector getExctEng(String tableName, String config);
   Vector getRangeExctEng(String tableName, String config,double emin,double emax);
   Vector getRangeBndfree(String tableName, String config,double emin,double emax);
    ….
}
```

The function getAllElements returns information about how many data elements are available in the current database. This information appears in a combo-box in the graphic user interface. The function getPreAvgEng obtains the lowest 10 configuration average energies for each ion stage, as we discussed before. The functions getRangeExctEng and getRangeBndfree obtain the photoexcitation and photoionization data given an energy range. It is efficient when users only want to calculate the spectrum for a specified energy range.

For the web-based project MIXOPA, we use the Enterprise Java Bean (EJB) framework, which has similar functionality to CORBA, that is, they both provide a middle-ware that allows distributed objects to communicate with each other across different machines. Implementation of the EJB has some differences from implementation of CORBA. To implement an enterprise Java bean, two interfaces and one or two classes should be defined, that is, the remote interface, the home interface, the bean implementation class and the deployment descriptor. The definitions of these interfaces and bean classes are shown in Section 2.2.2. Here we give the remote interfaces used in project MIXOPA. The purposes of the functions defined in the interface are easily seen from the function name.

Table 4.3.3 The EJB interface used for project MIXOPA

```
public interface MixopaEJB extends EJBObject {
    public Vector getPhotonGrid()  throws java.sql.SQLException, RemoteException;
    public Vector getEOS(String element, String eos)
                    throws java.sql.SQLException, RemoteException;
    public Vector getMixedEOS(String[] elements, double[] fractions)
                    throws java.sql.SQLException, RemoteException;
    public OpacityDataSet getMixedOpacity(String[] elements, double[] fractions)
                    throws java.sql.SQLException, RemoteException;
    …
  }
```

4.4 Graphic User Interface Design

As we discussed in Section 1.1, the calculation of atomic data and opacities involves many control parameters input by users and needs a long batch job script that must be edited correctly before the calculation. This process is tedious and error-prone. To help users generate the atomic data for ICF applications, we built graphic user interfaces (GUIs) to provide a computing environment that integrates the specification of computing model, the atomic data calculation, the spectrum and opacity calculations and the graphic data presentation into a single platform. Using this tool, users can easily obtain almost all needed atomic data for ICF applications.

The interfaces are designed to be as simple as possible. Default values are predefined when these values are best parameters for the physical model. Users are also guided through the calculation process. For example, in the atomic data calculation module, after all data have been calculated for all ion stages, a form automatically appears to allow users to specify the atomic model that will be used to generate the atomic data table. The primary layout of the GUI written in Java is shown in Fig. 4.4.1.

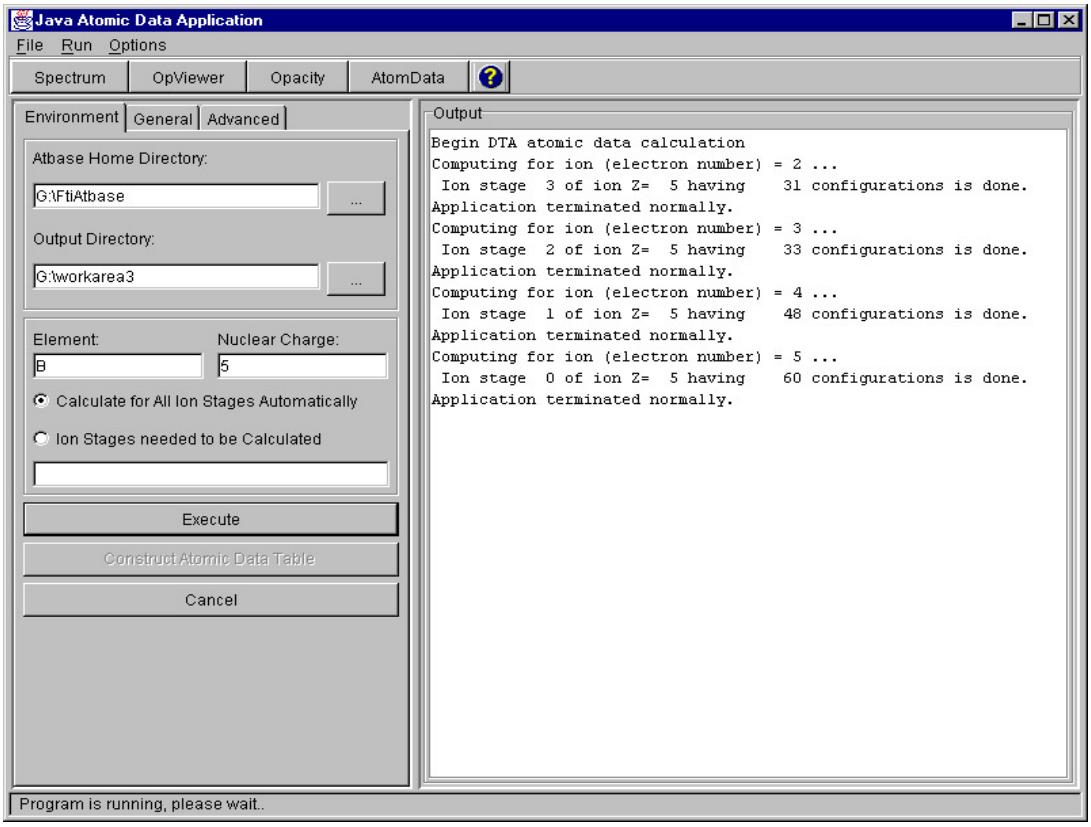


Fig. 4.4.1 The graphic user interface for atomic data computing

As we can see, there are four components in the project: the atomic data calculation, the opacity computation, the spectrum analysis and the EOS and Opacity visual tool. The atomic data calculation module can be used to calculate the atomic data under the DTA model for elements from Z = 1-18 and the UTA model for elements from Z = 1-79. For the DTA model, we can also calculate the atomic data for both LTE and non-LTE plasma conditions. The non-LTE atomic data include various rate coefficients such as dielectronic recombination, 3-body recombination. We use ATBASE as the underlying computing program. However, modifications are made so that these native Fortran programs can connect with the Java program. A sample screen shot for the atomic data calculation is shown in Fig. 4.4.1.

The opacity computation module calculates the EOS and opacity data or mixed opacity using the DTA or UTA atomic data models under the LTE approximation. Non-LTE opacities can also be obtained under the DTA model for elements ranging from Z = 1 to Z = 18. In the DTA opacity calculations, the program can automatically switch to non-LTE opacity calculation option if the plasma condition is not suitable for the LTE model. The criterion of plasma condition that the LTE assumption can be applied may be written as,

$$n_e \geq 1.6 \times 10^{12} T_e^{1/2} I_{mn}^3 (cm^{-3}),$$

where I_{mn} is the excitation potential for the transition from n state to m state.

Users can easily generate an EOS and opacity table for hydrodynamic calculations. The output table can be directly used by the BUCKY radiation

hydrodynamics code. The program also provides flexible controls in the EOS and opacity table generation, such as the temperature and density grid, and the photon energy boundaries. A sample screen shot for the EOS and opacity calculation is shown in Fig. 4.4.2.

Implementations for these two modules are similar. A common abstract panel is defined as containing three main parts: the first one is the panel used to construct the computing task. Users give the calculation parameters and specify the computing model in this panel. The second is the output window which gives the calculation results and the computing process information. The third is the program status bar. The panel of the atomic data calculation module and the panel of the EOS and opacity calculation module inherent from the abstract panel but implement their own functions. In order to avoid the freezing of the interface when executing the calculations, a separate thread is spawned to handle the calculation. The thread interruption mechanism is also applied to allow the underlying processing program to terminate gracefully.

After the EOS and opacity calculation, a menu of visual tools shows up. For a single temperature and density pair, users can view the results of ion stage populations, frequency dependent opacities and absorption spectrum. Users can study the opacity components that contribute to the total opacity. In Fig 4.4.3, we show the total opacity and its three components for Au plasma (T=100eV, D=$10^{21} cm^{-3}$), that is, the bound-bound, bound-free and free-free transitions.

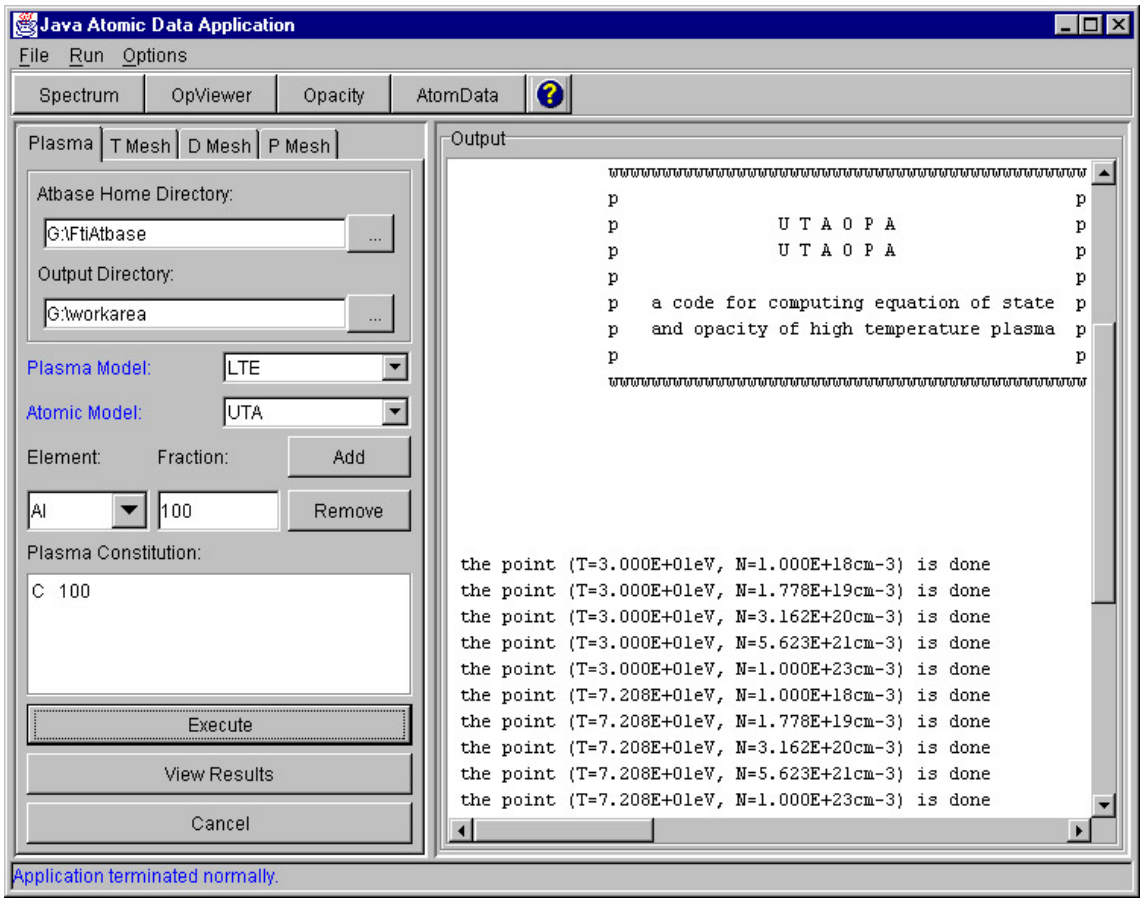


Fig. 4.4.2 The graphic user interface for EOS and opacity computing

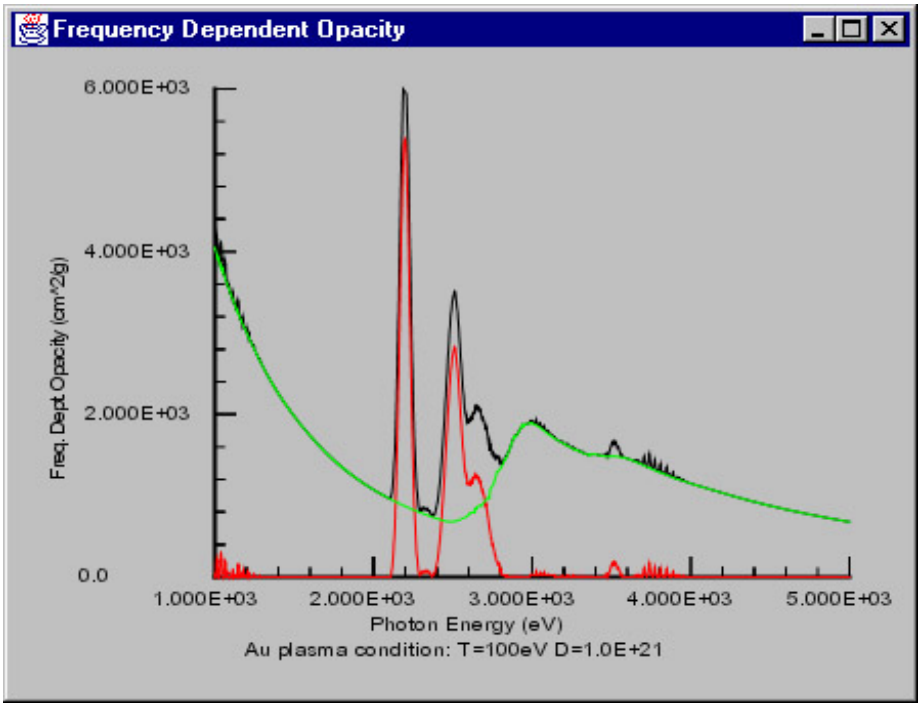


Fig 4.4.3 Sample screen shot of the total opacity and its three components for Au plasma (T=100eV, D=$10^{21} cm^{-3}$).

This module can also be used with the hydrodynamic application to analyze the plasma condition. For example, in Fig. 4.4.4 (a)-(d) we show a series of spectrum evolution under the change of plasma temperatures for Ar plasma. The density of Ar plasma is fixed at $10^{20} cm^{-3}$ and the plasma length is fixed at $10^{-3} cm$. The temperature changes from 25eV, 50eV, 75eV to 100eV. We can see the distinct spectrum changes with temperature, which therefore reflect the plasma condition. For multiple

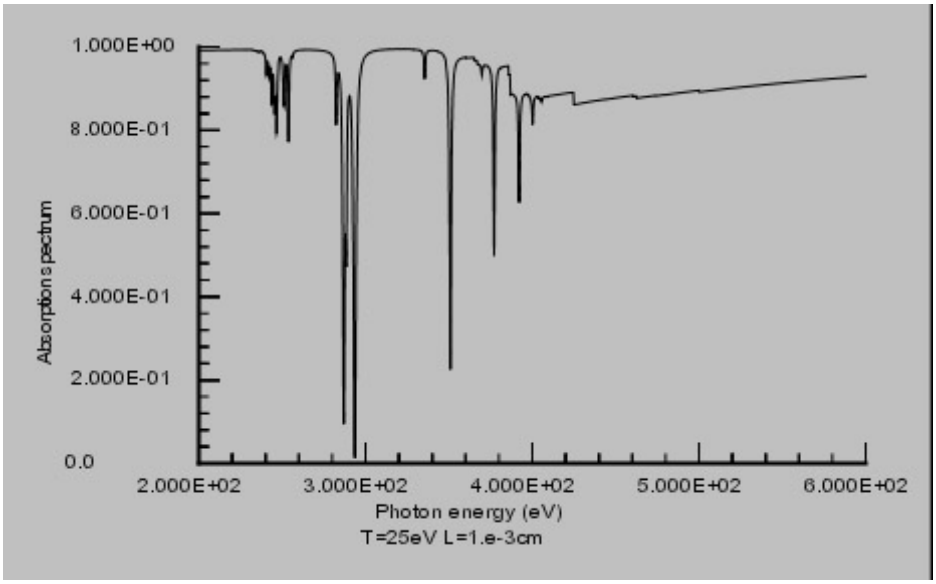


Fig.4.4.4 (a) Ar Spectrum (D =$10^{20}\,cm^{-3}$ T=25eV)

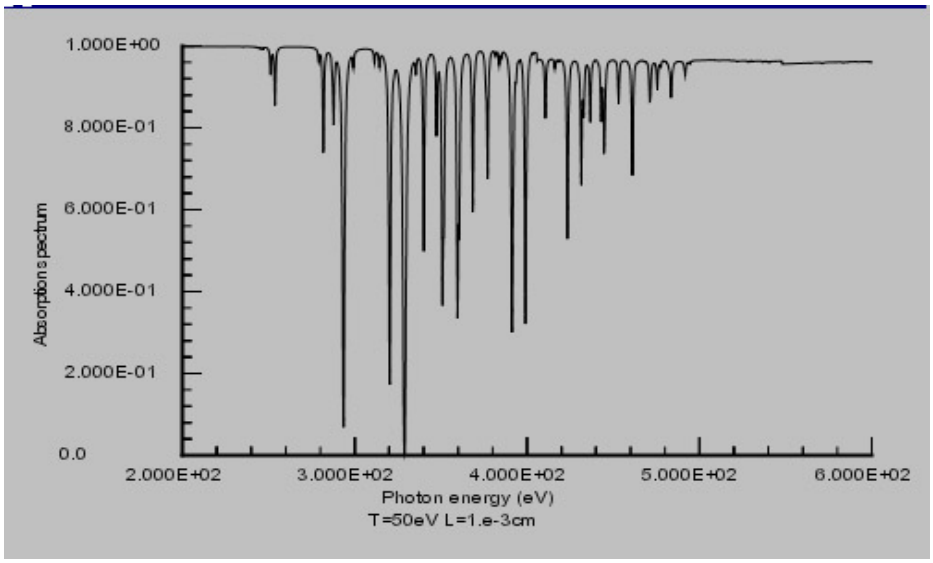


Fig. 4.4.4 (b) Ar Spectrum (D =$10^{20}\,cm^{-3}$ T=50eV)

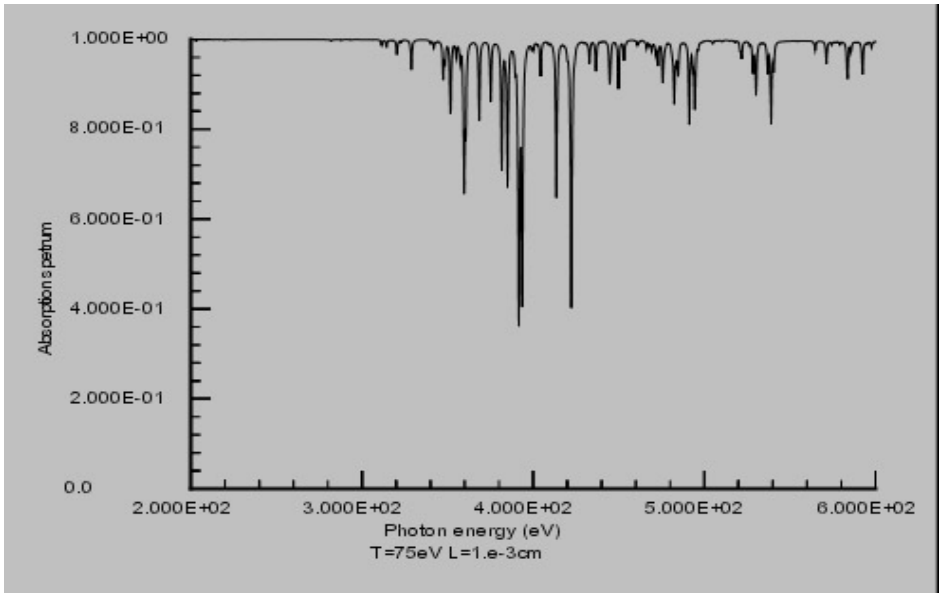


Fig. 4.4.4 (c) Ar Spectrum (D = $10^{20}\,cm^{-3}$ T=75eV)

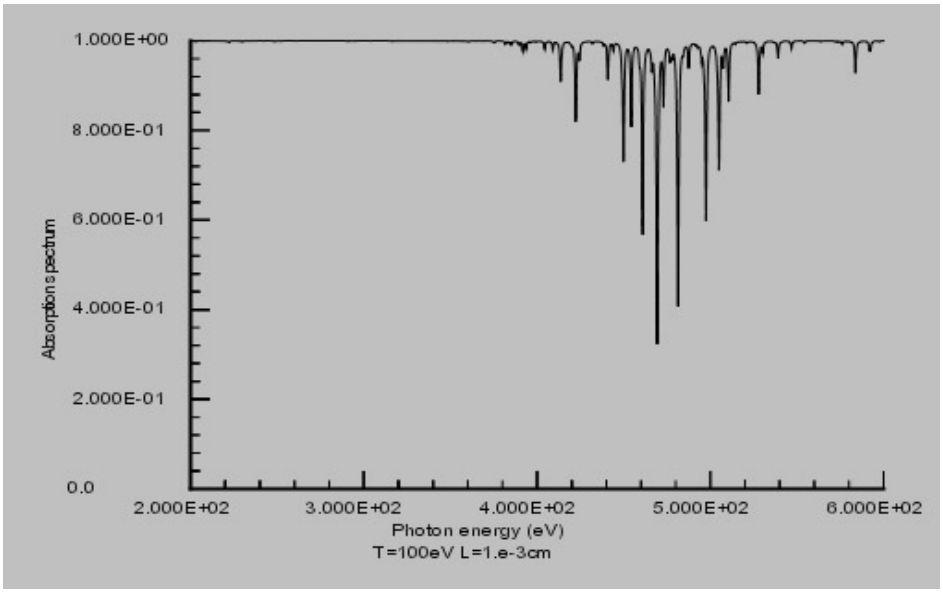


Fig. 4.4.4 (d) Ar Spectrum (D = $10^{20}\,cm^{-3}$ T=100eV)

.

temperature and density grids, the program automatically switches to the EOS and opacity viewer module.

The EOS and opacity viewer module is used to visualize the EOS and opacity results. It can run either as a standalone Java application or as a Java applet in a web browser. The data source can be from the local file system for Java application or from a remote data source for both Java application and Java applet. There are options for users to choose the part of the EOS and opacities to display. A sample screen shot of this module is shown in Fig. 4.4.5.

Implementation for this visual tool needs much more programming efforts than the previous two modules. The program dynamically determines the number of tabs and the drawing canvases according to the drawing options. In order to support multiple panels and reduce the user response time, each drawing canvas has an internal thread running for the data retrieve and data presentation. A drawing canvas has a maximize, a minimize and a close button just like a usual window. By right-clicking the drawing board, the pop up menu appears so that users can manipulate the graph or print the graph. To support the visual tool, we developed a Java 2D package (AtGraph2D). This package is built on the top of Sun's Java 2D package and it can be used as a generic Java 2D graph API. It provides almost all common graphic functionalities, such as the label definition, the title positioning, data line coloring, and so on. We describe the spectrum analysis module in the next section.

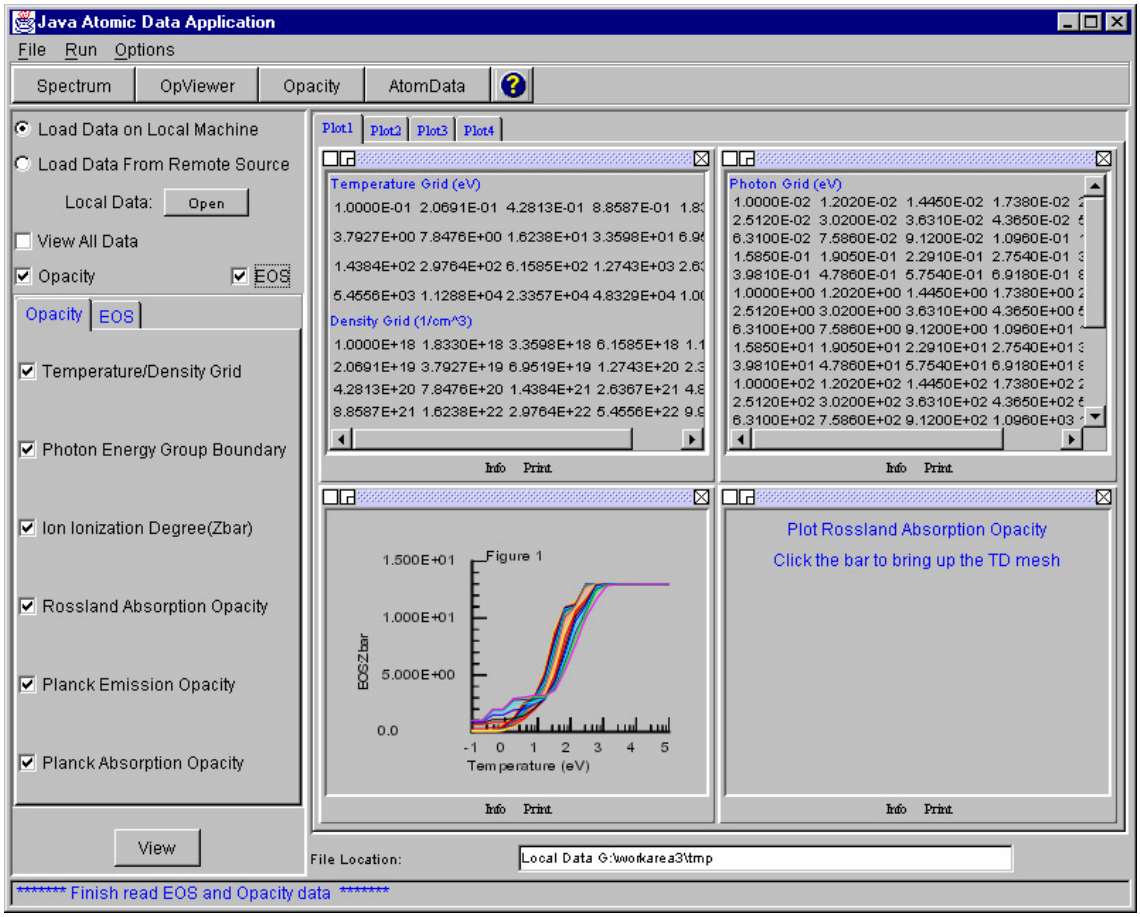


Fig. 4.4.5 The graphic user interface for the EOS and opacity visualization

4.5 Spectrum Analysis

The spectrum analysis module can be used to simulate the plasma spectra for low Z elements under the DTA model and for medium to high Z elements under the RSSUTA model. Its computing model employs the three-tier architecture, which involves three components: client, application server and database server. We use Oracle object-relational database management system as our database server, which has been discussed in Section 4.2. The application logic such as opacity, equation-of-state and spectrum calculation reside in the application server. JDBC is used as a protocol to connect with the database. The Object Request Broker (ORB) provides a transparent mechanism to invoke a method on server objects and establish the remote communication with distributed objects. The client uses the internet inter-ORB protocol (IIOP) to connect with the application logic and presents the results using our AtGraph2D graphic package. We have shown the three-tire architecture in Section 2.4

The graphic user interface includes three panels: the left panel is used to specify the plasma condition, the right panel is used as the main plotting area and the bottom panel is used to view the program status and to control the spectrum drawing process. The plasma condition specification includes the plasma model, the atomic model, the element which is from the list of all available elements determined by the database, the plasma temperature and density. Currently, only RSSUTA atomic model is implemented. The user can also specify the interested spectrum range in either

Angstrom or eV units and choose the electron configurations that will be included in the simulation. The right window is implemented as a container, which can contain as many results as necessary for different plasma conditions. The spectrum for each plasma condition is shown on its own panel. The panel inherits the generic AtGraph2D class (as discussed before) which provides some basic functions. Besides these basic functions, the panel can read external data, for example, experimental data into the graph for comparison (as shown in Fig. 4.5.4 and Fig. 4.5.5). A sample screen shot of this module is shown in Fig. 4.5.1. This figure shows two simulations: one is for Ge plasma (Z=32), the temperature is 76eV and the ion density is $2\times10^{20}/cm^3$, the other is for Nb plasma (Z=41), the temperature is 47eV and the ion density is $10^{20}/cm^3$. We give more detailed comparison with experiments later.

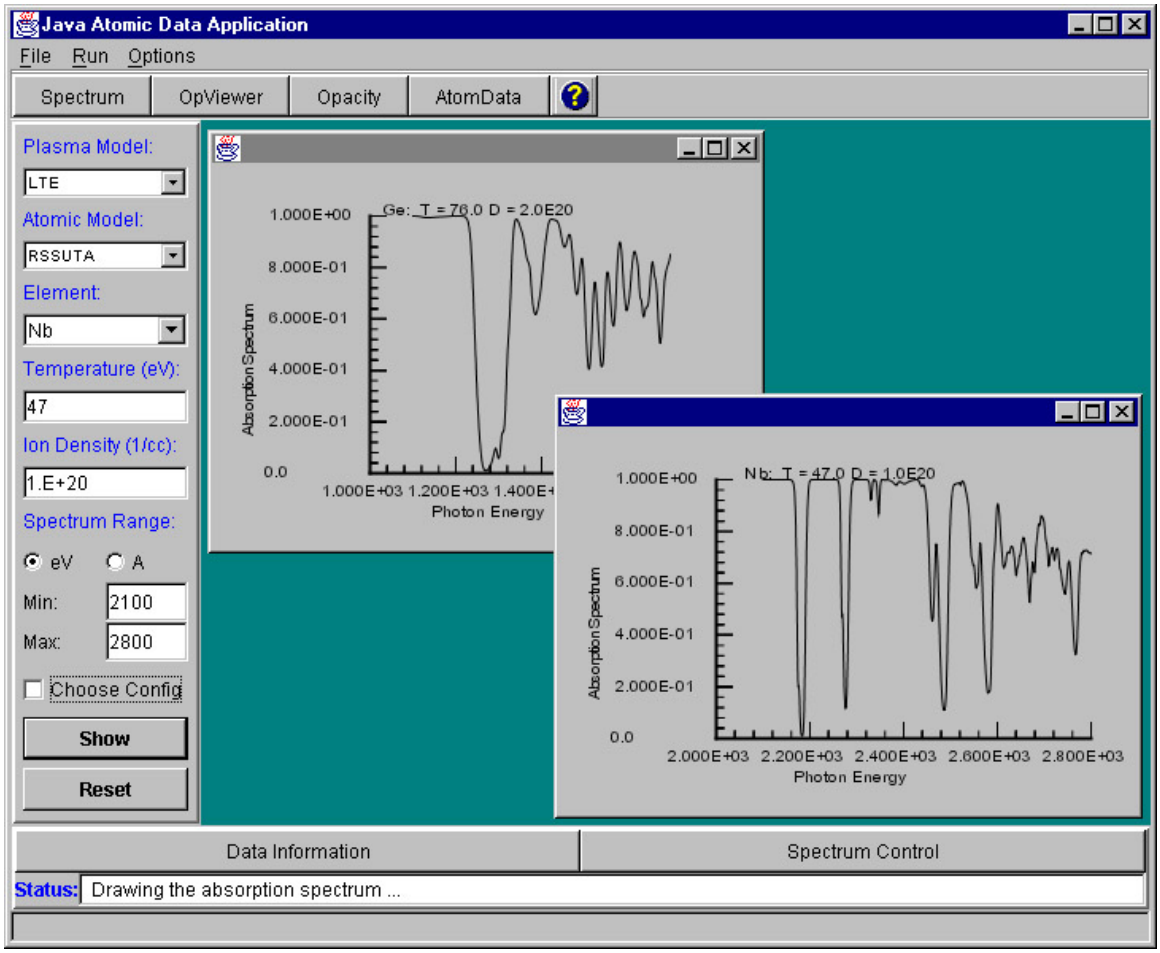


Fig. 4.5.1 The graphic user interface for spectrum analysis

The spectrum analysis calculation process begins after the user clicks the "Show" button. The program first loads the relativistic configuration average energies from the database if needed data are not in the data cache, and solves the Saha equation to obtain the configuration populations, then determines the important configurations that have populations above a certain criterion. For these important configurations, the program loads the photoexcition and photoionization data from the database and caches those data for the next run if the user is not satisfied by the current results. Finally, the program constructs the spectrum and the results appear on the right panel. A detailed description about the program and database interaction has been given in Section 4.3.

A powerful feature of the program is that users can access progressively more detailed data information, such as the ion stages, configurations which are important in the calculation and configuration populations, the transition arrays involved in the spectrum, the photoionization edges and so on. This is very convenient for users to understand the spectrum structure. Figure 4.5.2 shows a sample screen shot of the detailed transition lines from a Ge absorption spectrum simulation. Using the spectrum control, users can also plot the individual contribution from each ion stage. Sample screen shots for showing the individual contribution from each ion stage can be found in the User's Manual.

To test the accuracy of this program, we compare our simulation results with some available experiments.

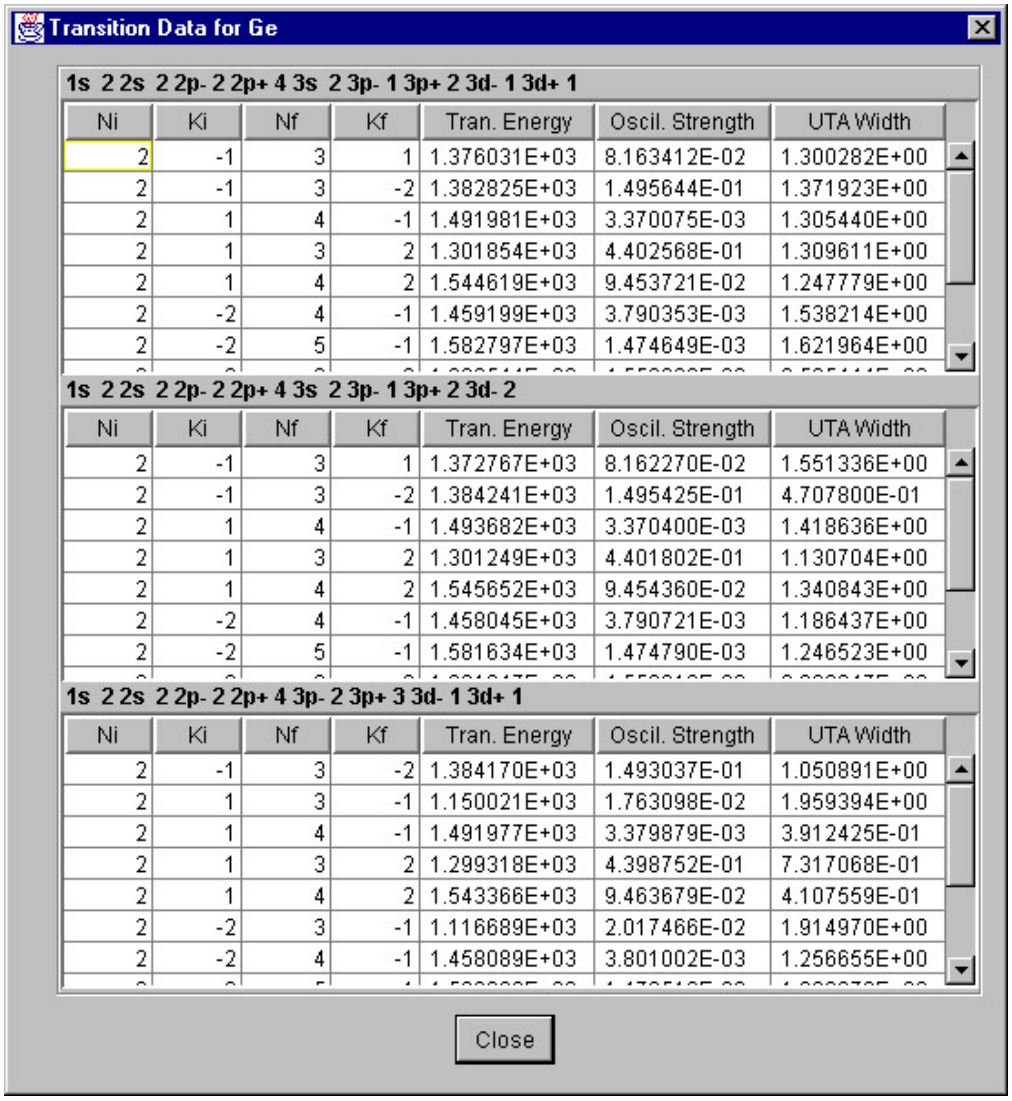


Fig. 4.5.2 The graphic user interface for detailed data information

Comparison with experiments

Experiments that can be used as benchmarks for validating our opacity model are difficult to conduct. Because opacity models depend critically upon plasma conditions,

benchmark experiments require complete and precise characterization as well as precise transition data. The point projection spectroscope technique (PPS) has been extensively applied for this kind of experiment. The plasma to be studied is created either by direct or indirect irradiation. An auxiliary point source plasma, whose dimensions are small compared to the main plasma expansion, simultaneously generates an x-ray source which probes the main plasma along the target surface. The attenuation of the x-ray probe through the expanding plasma is measured with a space resolution on the order of the point source diameter and with a time resolution given by the duration of the point source plasma. Absorption data can be obtained for different time delay between the main laser and the diagnostic beam. The typical experiment setup used to measure the absorption spectrum of a radiatively heated plasma is shown in Fig. 4.5.3 [MCK95].

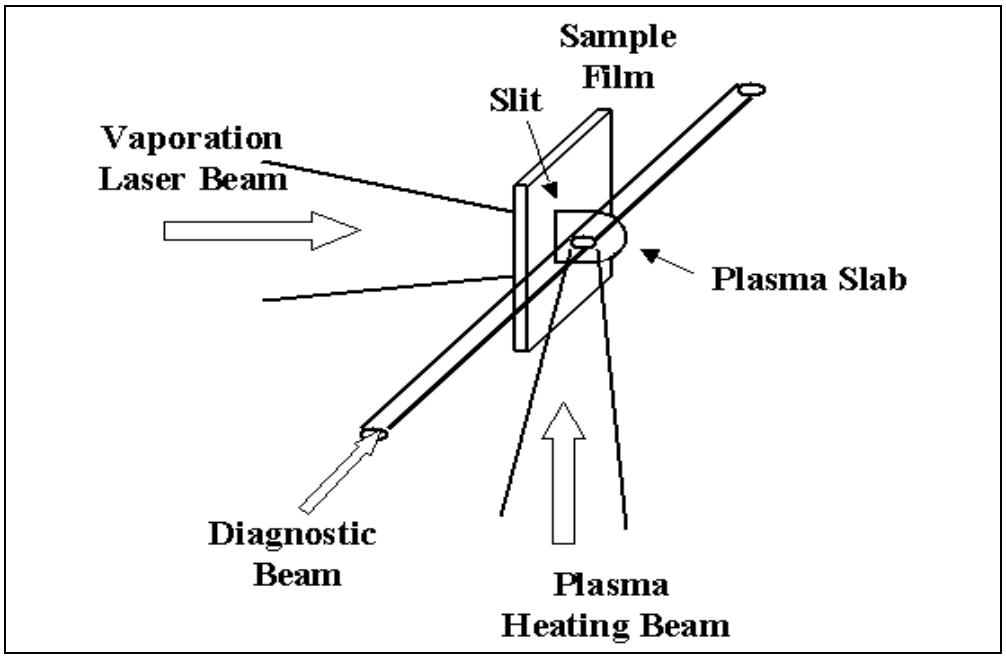


Fig. 4.5.3 Schematic of the typical experimental arrangement used to measure the absorption spectrum of a radiatively heated plasma

Ge absorption spectrum

Reference [FHSR91] presented the transmission spectra of a Ge plasma at T=76eV and an ion density of $0.05\,g/cm^3$. Thin foil samples are indirectly heated using thermal x radiation from separate laser-produced plasmas created by focusing the two main beams of the HELEN Nd-glass laser onto a gold target. Each beam delivering up to 220J of energy at 0.53μm wavelength in a 200ps duration pulse. The sample foil contains 3000Å thickness of germanium which is tamped by 1μm thickness of Parylene-N on both sides. The experimental data has been analyzed in previous publications [FHSR91][GBD95] . We also take this example to test the accuracy of our model. For the experimental Ge

plasma condition, our predicted mean charge state $\overline{Z}$ from the Boltzmann-Saha equation is 14.15. There are 274 significant relativistic configurations and 7677 transitions involved in the calculation. Figure 4.5.4 shows the Ge transmission spectra between 1100eV and 1700eV from experimental measurement and the numerical simulation of the RSSUTA model. Experimental data is read from an external data file. The detailed spectrum information can be fairly easily obtained by querying the database. We can easily identify the prominent features, that is, the spectra at hν = 1300eV, hν = 1350eV and hν = 1500eV belong to the 2p-3d, 2s-3p and 2p-4d transition arrays, respectively. We can also view the separate contribution from an individual ion stage or from photoexcitation and photoionization by using the "Spectrum Control" button. The agreement between the RSSUTA calculation and experiment is very satisfactory.

Nb absorption spectrum

Perry *et al.* [PDS91] have reported their absorption measurements on x-ray heated niobium targets doped with aluminum as a temperature diagnostic. This experiment also used the point projection spectroscopy (PPS) technique. The simulated target consists of two 1500Å thickness layers of CH placed on either side of a 3400Å thick Nb element layer. The experiments provide benchmark data for LTE opacity codes for the moderate-Z element (Nb). Three different codes HOPE [RL90], ENRICO [WAL91] and STA [BOG89] were compared with experiment data. All three codes have the broadly correct spectral character. In this thesis, we also present our results for this case. The plasma condition in our computation is the same as in the experiments at a temperature of 47eV

and a density of $0.026\,g/cm^3$. The ionization balance was calculated as 11.86 using the Boltzmann-Saha equation. The results for $\overline{Z}$ from HOPE, STA and ENRICO calculations are 11.37, 11.75 and 12.15 respectively. We show the transmission spectrum for the measured spectral range in Figure 4.5.5, comparing with experiment data, which is read from Ref. [PDS91]. An instrumental profile width of 1eV is also convolved in our calculation. We can see that the shapes of the two spectra are very similar. The simulations are in good agreement with the measurements for $2p_{3/2}-3d_{5/2}$ and $2p_{1/2}-3d_{3/2}$ absorption peaks. For high photon energies such as absorption peaks $2p_{1/2}-5d_{3/2}, 2p_{1/2}-4d_{3/2}, 2p_{3/2}-5d_{5/2}$ and $2p_{3/2}-4d_{5/2}$, we are required to shift the calculated peak by about -30eV in order to have a good agreement. The shift is also needed by the STA method to achieve better agreement [PDS91]. We find the main contribution to the 2p-3d transition is from the Ni-like(Nb XIV) and Cu-like(Nb XIII) ions, and again we can easily identify the important configurations using the data information tool.

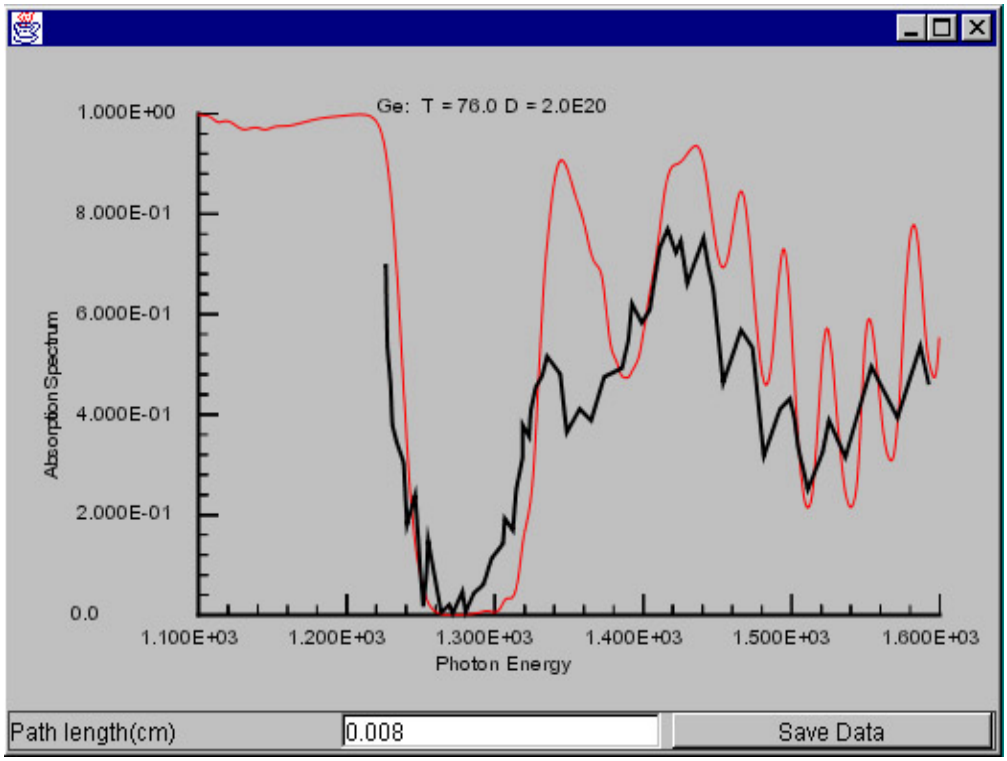


Fig. 4.5.4 Ge transmission spectra between 1100eV and 1700eV from experimental measurement [FHSR91](black line) and the numerical simulation of the RSSUTA model (red line). The plasma condition is shown as T = 76eV and D = $2\times10^{20}\,cm^{-3}$. The plasma path length used in the simulation is 0.008cm.

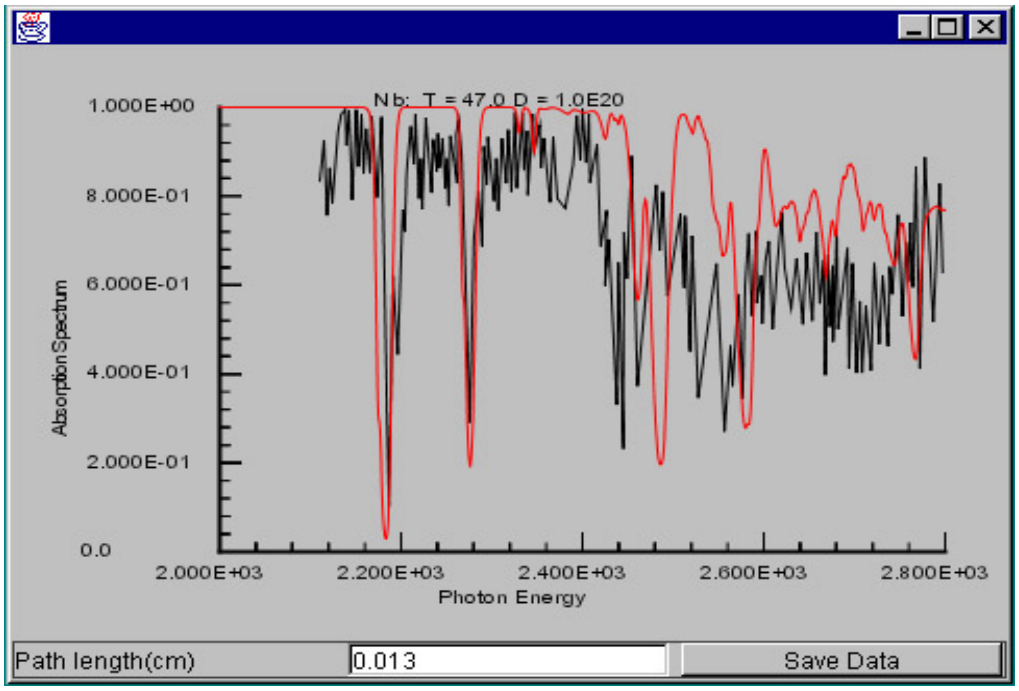


Fig. 4.5.5 Nb transmission spectra between 2000eV and 2800eV from experimental measurement [PDS91] (black line) and the numerical simulation of the RSSUTA model (red line). The plasma condition is shown as T = 47eV and D = $10^{20} cm^{-3}$. The plasma path length used in the simulation is 0.013cm. The energy positions for high energy transitions (2p-4d, 2p-5d) are needed to shift about 30eV in order to have a better agreement.

Au emission spectrum

The final example is the Au emission spectrum. In laser-produced plasmas, the x-ray spectrum of highly ionized high Z ions always shows the prominence of M-shell transitions of nickel-, copper-, zinc- and gallium- like ions. For example, in a gold plasma, x-ray line spectra of M-shell transitions in the wavelength range from 4.2 to 5.4 Å are important. This type of experiment is usually under non-LTE plasma conditions. The assumption of the detailed balance of collision and three-body recombination process is invalid. The dielectronic recombination may be the main process responsible for the population of the upper energy levels. Therefore, a full rate equation solution that includes various rate coefficients should be used. However, when the plasma condition is not very far from the non-LTE, we can still use a LTE plasma model to simulate the spectrum. The concept of an ionization temperature T_Z is introduced to simulate the non-LTE population distribution that is not too far from LTE, which is defined by $\overline{Z}_{LTE}(T_Z) = \overline{Z}_{non-LTE}(T_e)$ [BA86].

We computed the atomic data in the RSSUTA approximation for all 79 ion stages on the NPACI "Blue Horizon" IBM SP. The generated data size is about 215M bytes. All data are moved (ftp) from Blue Horizon (San Diego Supercomputing Center) to our local machine and stored in the Oracle database. Taking into account the disk space used by the DBMS internal management, the total disk space needed for storing the Au RSSUTA model data is about 300M bytes. The emission spectrum is calculated under the LTE condition. As illustrated in Ref. [BA86], the Au x-ray line spectra have

shown the prominence of M-shell transitions of nickel, copper, zinc, and gallium-like ions. The spectator electrons contribute the broad red wing. We show the comparison between our simulation in the graphic user interface and experiment in Fig. 4.5.6(a) and Fig. 4.5.6(b). Comparing with experiment, we found that when the ionization temperature T_Z is equal to 700eV and the density is $3\times10^{22}cm^{-3}$ (experiment density) the agreement of the spectral profiles between experiment and simulation is good.

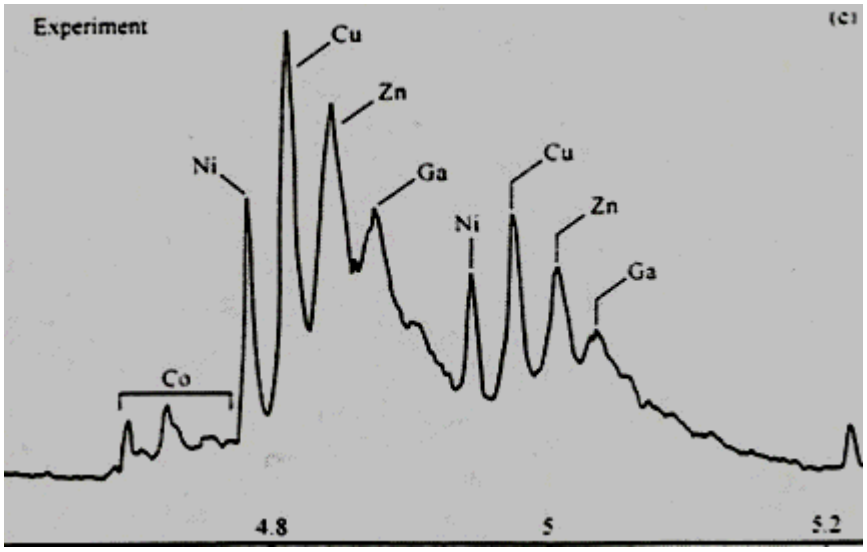


Fig. 4.5.6(a) Au emission spectrum for laser-produced plasma [BA86]

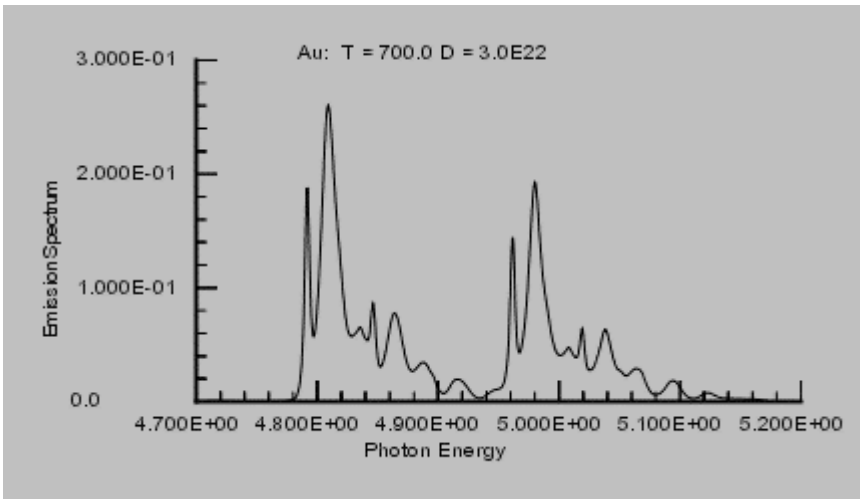


Fig. 4.5.6(b) Simulated Au emission spectrum under the RSSUTA model. The ionization temperature T_Z is equal to 700eV.

4.6 Web Target Projects

The above GUIs are implemented as standalone Java applications. However, for the applet to run within a web browser, the sandbox restriction prevents the connections to hosts other than the original one from where the applet is downloaded. Considering this restriction, the tasks that the Java standalone application and the Java applet can perform are different. The Java application can do the atomic data distributed computing in addition to the database query and opacity calculations. It can calculate all the basic atomic data using a computer cluster and put the data into the database. However, users who access through the Java applet can not perform the fundamental atomic data computing. One reason is performance concern. Because the calculation of atomic data may need several hours, it is not reasonable to let an applet run in a browser for hours. The other and the most important reason is the security issue. Therefore, we only provide three programs that are web-based. The EOS and opacity visualization module is adapted to run as a Java applet without much programming effort. This tool is for web users to visualize the EOS and opacities stored in our database. Another tool, which is used to generate the electron configurations, is also designed as both a Java application and a Java applet. The MIXOPA program uses Java servlet technology to calculate the mixed opacity based on the linear interpolation method. The underlying network architecture is shown in Fig. 2.4.2, in which the Oracle Application Server provides a middle-ware between the user and the database.

Project MIXOPA employs the Java servlet web programming technology. Several Java servlets, which implement the mixed opacity calculation procedure, are deployed in the application server. The linearly mixed EOS and opacities mean that all mixed data are calculated by the weight of the element fraction. For example, the mixed data for C_2H_8 are equal to the summation of 1/5 C and 4/5 H. The web page used to do the mixed opacity calculation is shown in Fig. 4.6.1.

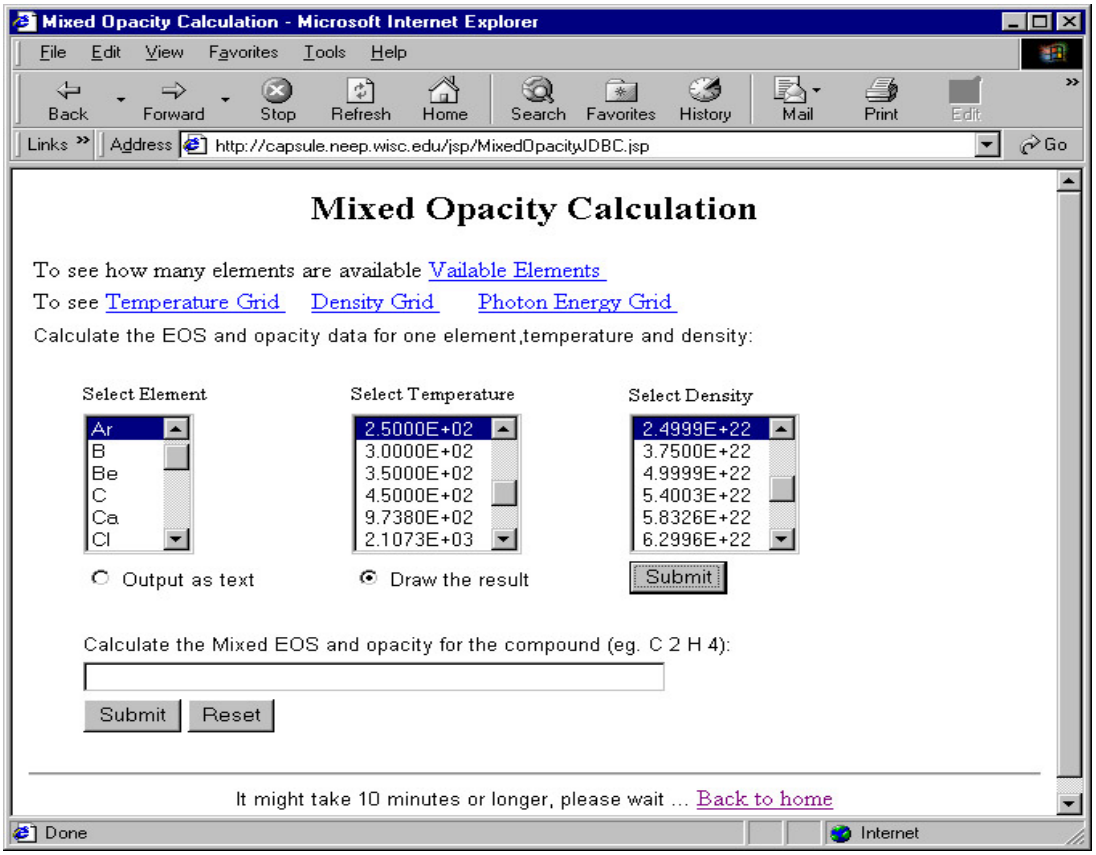


Fig. 4.6.1 The web page used to the mixed opacity calculation

Project ECGEN generates all possible electron configurations for a given number of electrons. The following screen shot shows the graphic interface for this program. After users give the total electron number and specify how these electrons occupy the principal and the angular quantum states, all possible electron configurations are generated and presented on the right panel. The usage of this program can be found in the User' manual.

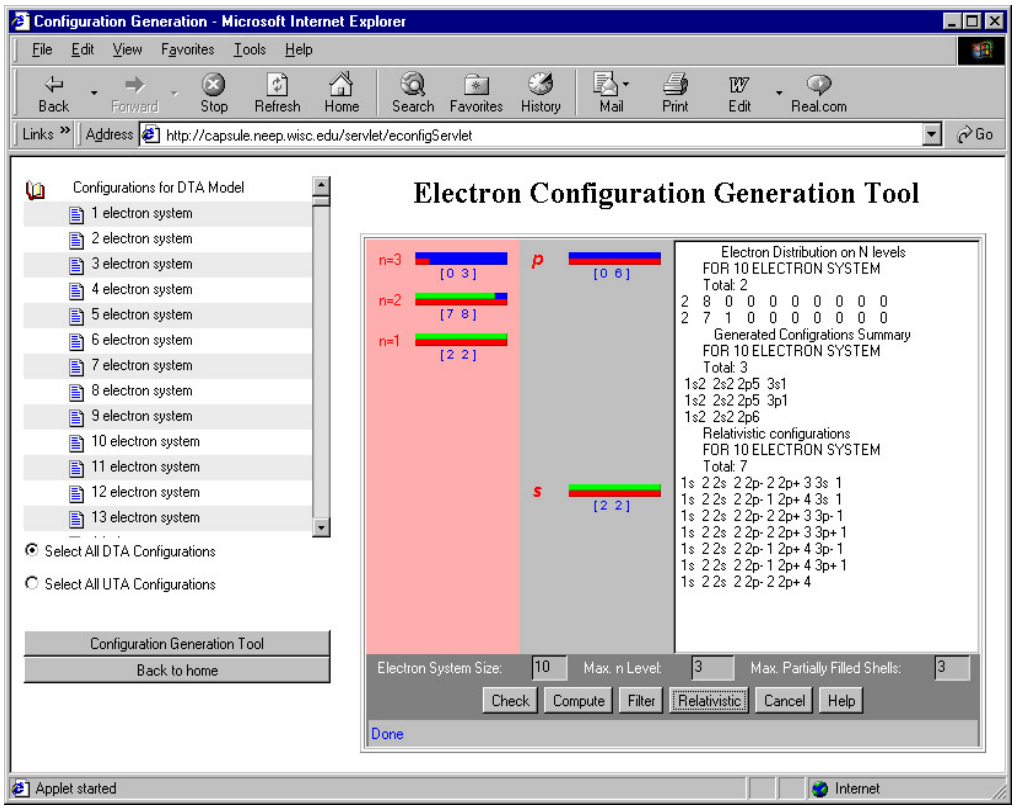


Fig. 4.6.2 Graphic user interface for generating electron configurations

Chapter 5

Conclusion

As the result of cross-disciplinary research, this thesis involves three areas: atomic physics, computational science and information technology. Conclusions will be made according to these three areas as following:

I. Atomic Physics Aspect:

A primary goal of the thesis is to provide a practical user-friendly environment that allows users to generate large scale and high quality atomic data for ICF research applications as well as do spectrum analysis for medium to high Z elements. To achieve this goal, the computing system integrates the modification of an existing atomic data calculation suite ATBASE [PW96] and the RSSOPA parallel program together. The major features of this atomic data computing program are:

- **Atomic data under DTA and UTA models**: Atomic data can be calculated under either the DTA model (Z = 1-18) or the UTA model (Z = 1-79). For the DTA model, atomic energy levels, oscillator strengths and photoionization cross sections are calculated using the non-relativistic Hartree-Fock method with options of multi-configuration interactions and relativistic correction. Comparing with experiments, the accuracy of the data is quite good. For the UTA model, the average configuration

approximation is used under either the non-relativistic formalism (ATBASE) or the relativistic formalism (RSSUTA). Because of its relativistic treatment, the RSSUTA model is more accurate than the ATBASE UTA model. Various rate coefficients can also be calculated for non-LTE plasmas (Z = 1-18) under the DTA model.

- **EOS and opacity data for ICF applications**: The atomic data generated by the atomic data calculation module are applied automatically to compute the EOS and opacity under either the LTE plasma condition or the non-LTE plasma condition. For the LTE condition, current supported models are the UTA atomic model for Z = 1-79 and the DTA atomic model for Z = 1-18. For the non-LTE condition, the supported model is the DTA atomic model for Z = 1-18. For group opacity, the photon group structure can be automatically setup or be defined manually by advanced users.
- **Spectrum analysis considering the JJ coupling**: Our RSSUTA model uses the fully relativistic JJ coupling schema. We show the importance of the JJ coupling in the UTA calculations by illustrating the spectra patterns for transition lines from low Z to high Z elements. The effect of JJ coupling on opacity and spectrum analysis can be significant when comparing with experiments. Transition arrays can be split into subarrays in the JJ coupling, which give more detailed spectrum structure than in the LS coupling. Compared with the STA theoretical model and experiments, the accuracy deviation of the RSSUTA model is within a very low percentage (under 5%).

II. Computational Science Aspect

- **Implementation of the RSSUTA model in parallel computing environment using MPI**: Large scale atomic data generation needs high performance computing technology. To calculate enormous numbers of transition lines and other radiative properties for high Z elements, we implement our RSSUTA model in distributed memory parallel fashion. In order to minimize the communications cost, the relativistic single-configuration single-electron transition model is applied to ensure the electron configuration independence. Experimental runs on the NPACI Blue Horizon IBM SP shows that the speedup reaches the ideal situation, that is, the speedup is linearly dependent on the number of processors.
- **Distributed computing using CORBA**: In order to connect the database and the front-end applications, the CORBA middle-ware technology is used to provide an object oriented distributed computing framework. The language neutral property of CORBA allows the integration of legacy systems implemented in other programming languages. The decomposition of application functionality into separate presentation, application and data services results in a distributed computing architecture for computational grids. Another similar technology Enterprise Java Bean (EJB) is also applied for projects in which only the Java programming language is used.

III. Information Technology Aspect

- **Atomic database development**: We use the Oracle database management system to store the atomic data and we focus on the atomic data required by Inertial Confinement Fusion (ICF) applications, which is the opacity data and spectrum data. For opacities, we currently store the results of EOS and opacity data for elements Z = 1-20 into the database. These data are used as basic data in the mixed opacity calculations. For spectra, we store radiative properties such as transition energy, oscillator strength, UTA width into the relation-object tables in the Oracle database. A Saha equation solver has been developed to use the atomic data retrieved from the database to calculate energy level populations. These data provide detailed information in the spectrum analysis.
- **Graphic user interfaces**: For general users in large scale atomic calculations, one of the difficulties is the preparation of input parameters. We have developed four major packages to solve the problem: atomic data calculation module, EOS and opacity calculation module, high-Z spectrum analysis module, and EOS and opacity visualization module. These modules provide graphic user interfaces from the data generation to the data visualization. Using these programs, users can easily obtain almost all atomic data for ICF applications. Other tools such as the Java graphic 2D package (ATGraph2D) and electron configuration generation tool have also been developed in the program.

IV. Further considerations

- **Non-LTE high-Z opacities**

  LTE opacities are valid in cases that collision processes are dominant. In ICF research, plasmas are often in non-LTE conditions and opacities for non-LTE conditions are very important. Although our current program can calculate non-LTE opacities for elements Z = 1-18, it can't calculate non-LTE opacities for high Z elements. There are two kinds of difficulties in this situation. One is the lack of systematic theory for calculations of various rate coefficients. Although there are existing empirical formulas for three-body collision/recombination, electron collision excitation/deexcitation, it is difficult to calculate the dielectronic recombination because of a number of intermediate states involved in the process. Approximations are usually used in the dielectronic recombination coefficient calculations. For high Z elements, the other difficulty is the need of enormous computing efforts.

  The thesis shows the successful use of the three-tier computing architecture in the atomic data and spectrum analysis. We have shown under the RSSUTA model the computing speedup is linearly dependent on the number of processors. It is possible we can use this approach in the high-Z dielectronic recombination rate coefficient calculations. The database can be used in the high-Z opacity calculations to serve as a better data provider. The capability of extending the current computing system to high-Z non-LTE opacity is worth further study.

- **Improvement on the current system**

  Several improvements can be made to the current atomic data computing system. First, the SDSC grid portal toolkit (GridPort) [TMB00] can be used to connect with our applications so that users could invoke the computing program on IBM SP machines. Currently, users have to physically login to IBM SP machines in order to run the RSSUTA data calculation and transfer the calculated data to local machines by ftp. Second, the current system is based on CORBA distributed computing architecture. It may be possible to migrate the program to computational grid architecture in order to improve the performance.

Bibliography

[BA86] C. Bauche-Arnoult, *et al.,* ***Phy. Rev. A* 33**, 791(1986).

[BBK85] C. Bauche-Arnoult, J. Bauche and M. Klapisch, ***Phys. Rev. A* 20**, 2424(1979), ***Phys. Rev. A* 25**, 2641(1982), ***Phys. Rev. A* 30**, 3026(1984), ***Phys. Rev. A* 31**, 2248(1985).

[BC83] A. Burgess, M.C. Chidicjimo, ***Mon. Not. R. Astr. Soc***. **203**, 1269 (1983).

[BOG89] A. Bar-Shalom, J. Oreg, W.H. Goldstein, D. Shvarts, A. Zigler, ***Phys, Rev. A,*** Vol. **40**, 3183 (1989).

[BOG95] A. Bar-Shalom and J. Oreg, W.H. Goldstein***, Phys. Rev. E***, Vol. **51**, 4882(1995).

[BOS95] A. Bar-Shalom and J. Oreg, J.F. Seely, U. Feldman and C.M. Brown, B.A. Hammel, R.W. Lee and C.A. Back, ***Phys. Rev. E,*** Vol. **52**, 6686 (1995).

[BR83] Brian H. Bransden***, Atomic Collision Theory***, The Benjamin/Cummings Publishing Company, (1983).

[BS96] Lou Baker and Bradley J. Smith, ***Parallel Programming***, McGraw-Hill, (1996).

[CCW98] D.H. Cohen, H.K. Chung, P.Wang, J.J. MacFarlane, ***Atomic Radiation and Hydrodynamics Modeling in Support of the Sandia Light Ion Beam Program***, UWFDM-1074 (1998).

[COW81] R.D. Cowan, "***The Theory of Atomic Structure and Spectra***", University of California Press, (Berkely, California, 1981).

[DP82] M.W.C. Dharma-wardana and F. Perrot, ***Phys. Rev. A* 26** , 2096 (1982).

[EKK76] W. Ebeling, W.D. Kraft, D. Kremp, "***Theory of Bound States and***

Ionization Equilibrium in Plasma and Solids", Berlin: Akdemie, 1976.

[FHSR91] J.M. Foster, D.J. Hoarty, C.C. Smith, P.A. Rosen, and S.J. Davidson, S.J. Rose, T.S. Perry and F.J.D. Serduke, ***Phys. Rev. Let***, **67**, 3255(1991).

[FK97] I. Foster, C. Kesselman, ***Intl J. Supercomputer Applications***, **11**(2):115-128, 1997.

[FN65] U. Fano, ***Phys. Rev. A*** **140,** 67(1965).

[FS86] C.F. Fischer, ***Comput. Phys. Rep.*** **3**, 273 (1986).

[FTB94] C. Froese Fischer, M. Tong, M. Bentley, Z. Shen, and C. Ravimohan, ***J. of Supercomputing*** **8** , 117-134 (1994).

[GBD95] S. Gary, J. Bruneau, A. Decoster, *et al.*, ***J. Quant. Spectrosc. Radiat. Transfer*** **54**, 155 (1995).

[GLS99] William Group, Ewing Lusk, Anthong Skjellum, ***Using MPI***, The MIT Press, (1999).

[GMA86] J.C. Gauthier, P. Monier, P. Audebert, C. Chenais-Popovics and J.P. Geindre, ***Laser Part. Beans*****, 4**, 421(1986).

[GPR97] T.W. Gorezyca, M.S. Pindzola, F. Robicheaux, and N.R. Badnell**,** ***Phys. Rev***. *A* **56**, 4742 (1997).

[GRD99] I. Foster and C. Kesselman, editors. ***The Grid: Blueprint for a New Computing Infrastructure.*** Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1999.

[IRW92] C.A. Iglesias, F.J. Rogers, and B.G. Wilson, ***Ap, J***, **397**, 717 (1992).

[LEA00] Doug Lea, ***Concurrent Programming in Java***, Addison-Wesley, (2000).

[LG77] W.A. Lokke and W. Grasberger, ***LLNL Report***, UCRL-52276, (1977).

[LG96] Michael J. Lewis, Andrew Grimshaw, ***Proceedings of the Fifth IEEE International Symposium on High Performance Distributed***

Computing, IEEE Computer Society Press, CA, 1996.

[LLM88] Litzkow, M., Livny, M., and Mutka, M. W., ***Proceedings of the 8th International Conference of Distributed Computing Systems***, pp. **104-111**, June, 1988.

[LZ68] W. Lotz, ***Z. Phys***. **216**, 241 (1968).

[MCK91] A.N. Mostovych, L.Y. Chan, and K.J. Kearney, D. Garren, C.A. Iglesias, M. Klapisch, and F.J. Rogers, ***Phys. Rev. Let***, **75**, 1530(1995).

[MCW98] J.J. MacFarlane, D.H. Cohen, P. Wang, R.R. Peterson, G.A. Moses, C.A. Back, O.L. Landen, etc, ***Development of Soft X-ray Tracer Diagnostics for Hohlraum Experiments***, UWFDM-1069 (1998).

[MF89] J.J. MacFarlane***, Comput. Phys. Commun***, **56**, 259 (1989).

[MMS94] D. Mitnik, P. Mandelbaum and J.L. Schwob, etc, ***Phys. Rev. A***, Vol. **50**, 4911(1994).

[OHE96] Robert Orfali, Dan Harkey and Jeri Edwards, ***The Essential Distributed Objects***, (1996).

[PDS91] T.S. Perry, S.J. Davison, F.J.D Serduke, etc, ***Phys. Rev. Let.*** Vol**. 67**, 3784 (1991).

[PW93] P. Wang, ***EOSOPA, FTI Report UWFDM-933*** (1993);
P. Wang, ***Computation and Application of Atomic Data for Inertial Confinement Fusion Plasmas,*** UWFDM-855 (1991).

[PW96] Ping Wang, ***ATBASE User's Guide***, FPA-96-8 (1996).

[RL90] B.F. Rozsynai, M. Lamoureux, ***J. Quant. Spectrosc. Radiat. Transfer*** **43**, 381 (1990).

[RM99] Ed Roman, ***Mastering Enterprise JavaBeans and the Java 2 Platform***, Wiley (1999).

[RS92] S.J. Rose, ***J. Phys. B: At. Mol. Opt. Phys***. **25**, 1667 (1992).

[RZ72] B. Rozsnyai, Phys. Rev., A5, 1137 (1972).

[SN97] Pradeep. K. Sinha, ***Distributed Operating System Concepts and Design***, IEEE Press (1997).

[ST64] M.J. Seaton. ***Planet. Space. Sci***, **12,** 55(1964).

[TAN95] A.S. Tanenbaum, ***Distributed Operating System***, Prentice-Hall, Englewood Cliffs, NJ (1995).

[TMB00] M. Thomas, S. Mock, J. Boisseau, ***Proceedings of the ninth IEEE international symposium on High Performance Distributed computing***, P308, Aug. 2000.

[WAL91] B. Wilson, J. Albritton, D. Liberman, in ***Radiative Properties of Hot Dense Matter***, edited by W.H. Goldstein, C.F. Hooper, J.C. Gauthier, J.R. Seely, and R.W. Lee (1991).

[WOMG] Web site, ***http://www.omg.org.***

[WORC] Web site, ***http://www.oracle.com.***

[WSUN] Web site, ***http://java.sun.com.***

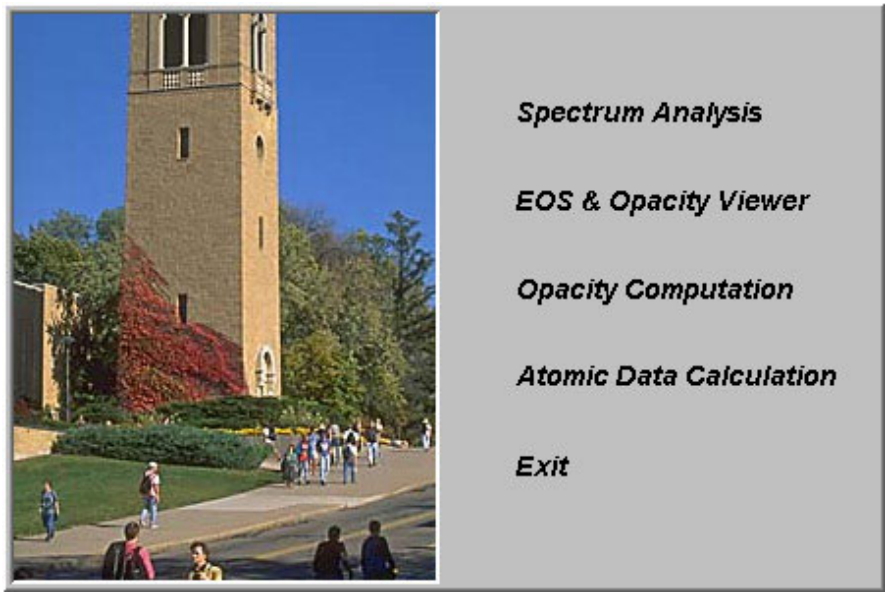
[YSZ96] J.K. Yuan, Y.S. Sun, S.T. Zheng, ***Phys. Rev. E* 53,** 1059 (1996).

Appendix A

High Performance Computation and Database of Radiative Properties with an Interface for ICF Applications

User's Manual

Jiankui Yuan and G. A. Moses



January, 2001

Fusion Technology Institute

University of Wisconsin-Madison

I. Electron Configuration Generation Tool

1. Program outline

This program can be used to generate electron configurations for opacity and spectrum calculations. It is designed to run as a Java applet in a web browser. It is written in JDK1.1.8 so that it can run in either Netscape or Internet Explorer without installing JDK1.2 plug in.

This program basically solves a linear equation with constraints. The total electron number is the right side of equation. The constraints of each variable are the possible occupation for each orbitor. The program can generate both non-relativistic configurations and relativistic configurations. Given the maximum allowed outer shells, the program can filter those configurations that users do not want.

Users first give the total number of electrons, then decide how many principle quantum levels those electrons can occupy. By right-clicking on the bar, users can set the minimum value and the maximum value for that N level. This gives a coarse refinement. To obtain a reasonable number of configurations, users usually need to specify the occupation information for the NL levels. By clicking the N level bar, the corresponding NL level bars will appear on the right side panel. User can also specify the minimum value and the maximum value for the NL levels by right-clicking on the bar. Then, by clicking the *Compute* button, the program first lists all combination possibilities for the N levels and the *Compute* button becomes the *Continue* button. By

clicking the *Continue* button, the program continues the calculation, spawns threads for concurrent processing for each possible combination. The final results will be shown on the right text area panel.

Sometimes, users may get configurations having too many outer unfilled shells. In this case, users can use the *Filter* button to filter out those configurations that have outer unfilled shells more than the maximum partially filled shells which is determined by the user. The corresponding relativistic configurations can also be generated by clicking the *Relativistic* button. All these configuration results are listed on the right text area panel. Users can obtain those configurations by copying and pasting to their text editors.

2. User Interface

In the following, some screen shots for generating configurations for 20-electron system are given.

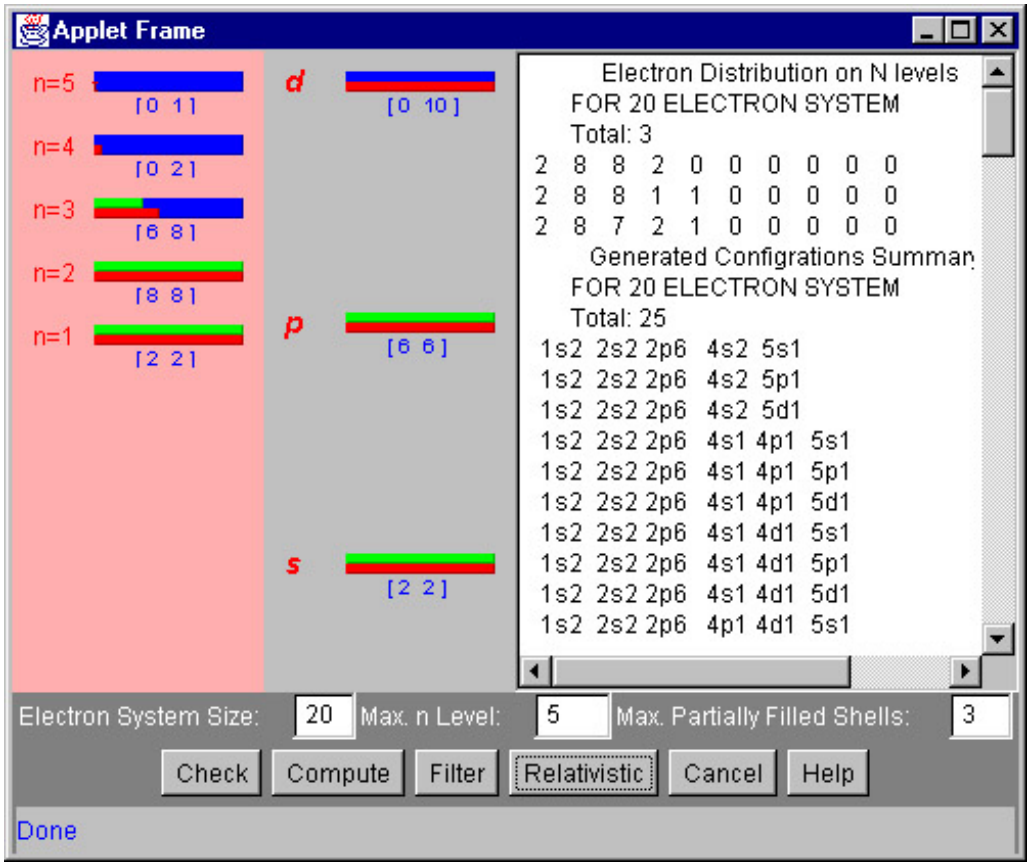


Fig. A.1.1 Non-relativistic configuration generation for the 20 electron system

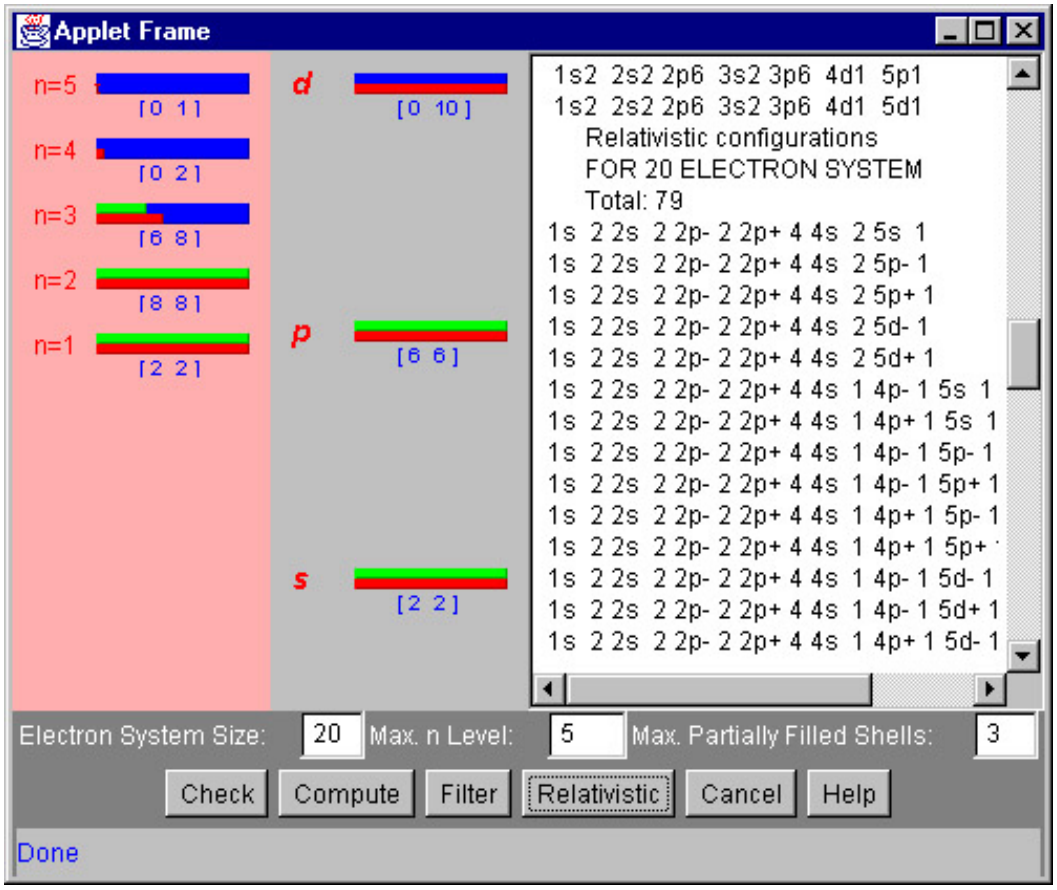


Fig. A.1.2 Relativistic configuration generation for the 20 electron system

II. Atomic Data Calculation

1. Program Description

This program is used to calculate atomic data under the Detailed Term Accounting (DTA) model and the Unresolved Transition Array (UTA) model. It is designed to make the calculation easier for general users. The underlying computing engine is based on ATBASE (see *ATBASE User's Guide (version 2),* Ping Wang, 1996). The user interface is programmed using Java. JNI (Java native interface) and other techniques are used to communicate with the native program ATBASE.

2. Executing Environment

Sun JDK1.2 or above are required. Microsoft Visual Studio 5.0 is used to compile and execute the FORTRAN code. The ATBASE programs include

1. xdtacfggen.exe: Generate the configurations for DTA model.
2. xutacfggen.exe: Generate the configurations for UTA model.
3. xatdata.exe: Calculate all atomic data such as energy levels and oscillator strength under both DTA and UTA models.
4. xatmodel.exe: Used by DTA model to apply different approximation to select atomic data.
5. xattable.exe: Construct final atomic data table used by DTA model.
6. xutatab.exe: Construct final atomic data table used by UTA model.

These programs have been tested successfully on Windows NT.

3. Program Outline

The atomic data calculation module provides a friendly user interface to invoke the native ATBASE program. Users specify the Atbase home directory so that the program can locate the executable programs and other parameters required. The output data will be written in the output directory. Then users input the element name and the nuclear charge they want to calculate. User can choose to automatically calculate all ion stages or some ion stages. This is useful because sometimes exceptions occur for some ion stages and these unsuccessful calculations must be repeated using alternative options. User also chooses the atomic model UTA or DTA. If user inputs the element nuclear charge above Ar for DTA model, the program gives a warning and does not execute. Parameters that control the wave equation algorithm solution and convergence are given as default.

The program automatically generates an input file based on user's specification and spawns a thread to handle the execution. The executing states and some debug output are shown on the right panel.

After all ion stages have been done, the button that constructs the atomic data table is enabled. By clicking the button, the program will automatically generate an

atomic data table if the UTA model is chosen. The UTA atomic data table files are uta.atomicdata and uta.photonizxx.

Under the DTA model, before generating the atomic table, users need to specify the atomic levels they want to include. A default-input form is popped up. After the user clicks the OK button, the program invokes the xatmodel.exe code to rearrange the raw atomic data. The construct table button will be enabled after the rearrangement. Then the program invokes xattable.exe to generate a DTA atomic data. A brief description about the number of energy levels included is shown on the right panel. The DTA atomic data table files are atomic.dat and pixfit.dat.

4. User Interface

In the following, some screen shots for the DTA and UTA calculations are given.

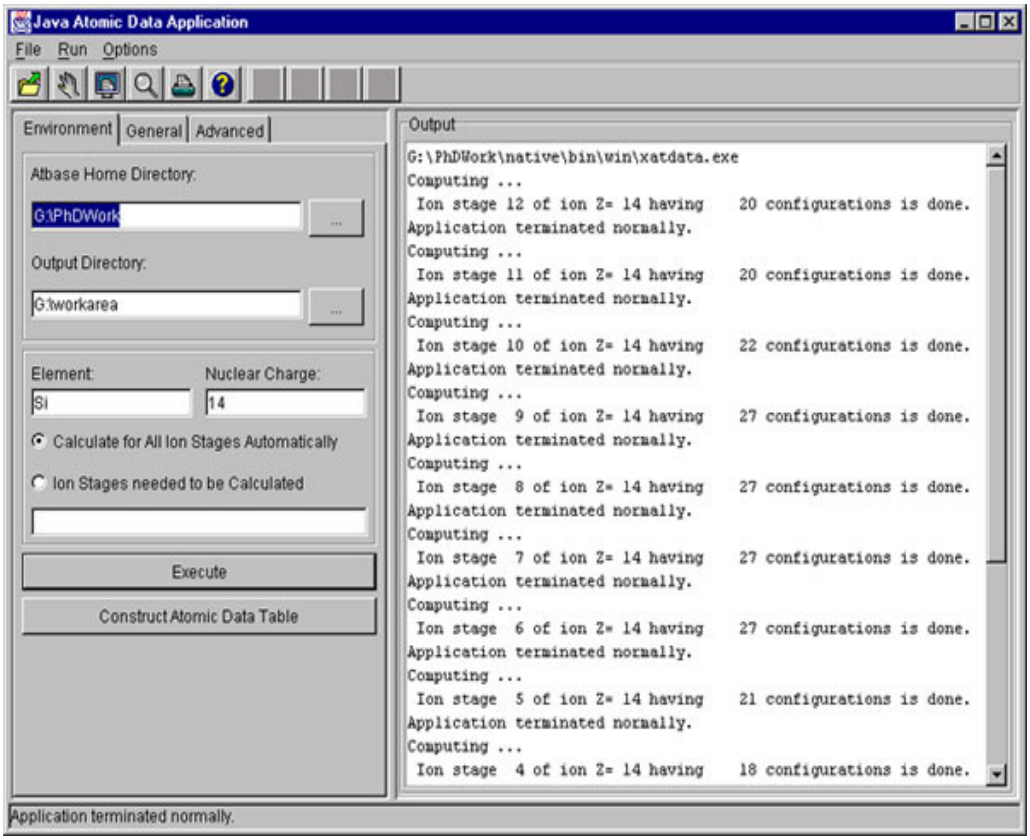


Fig. A.2.1 The environment tab for atomic data calculation

The environment tab contains information about the home directory and the work directory, the element specification and calculation options. The right panel gives debug output from the native code and Java code. Users can see the whole calculation process.

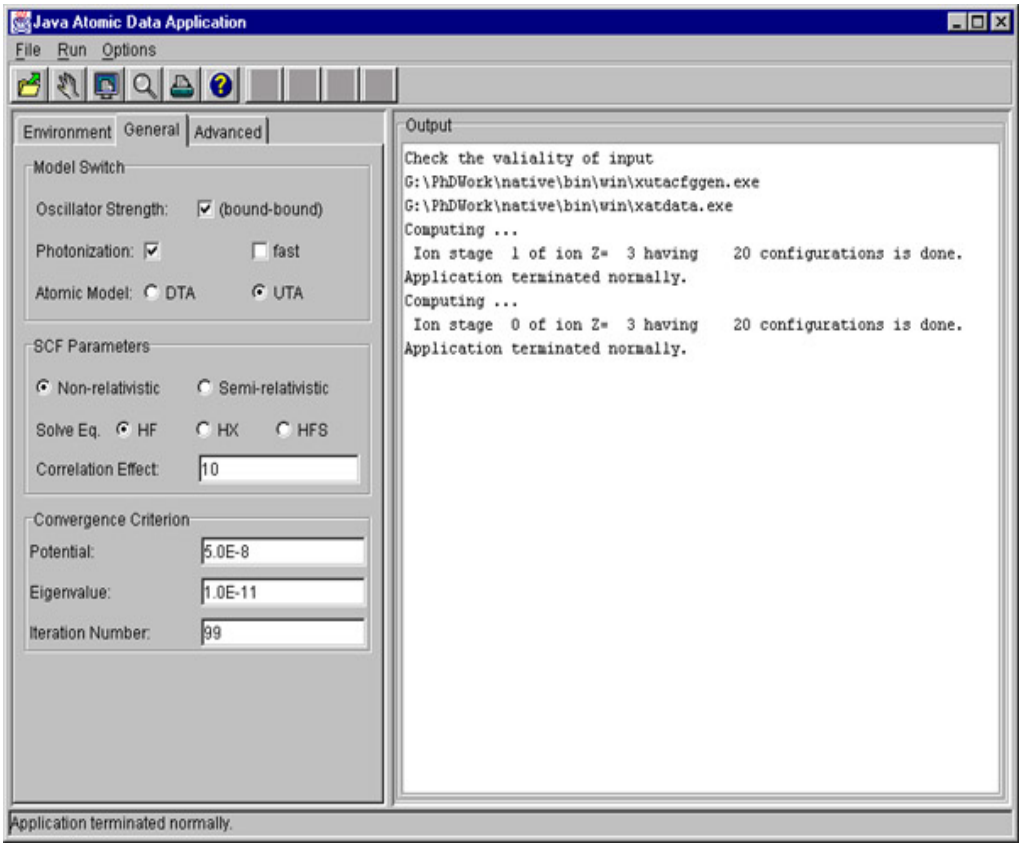


Fig. A.2.2 The general input parameter tab for atomic data calculation

In this tab, the user defines calculation control parameters. If the bound-bound checkbox is not checked, the bound-bound transitions will not be calculated. For high Z elements, the user can choose the fast way to calculate the photoionization cross section which uses the single electron single orbit wave function. The most important parameter is the atomic model.

The SCF parameters determine which potential form or algorithm is used in solving the Schrodinger equation. Other parameters can be left as default (shown in Fig. A.2.3).

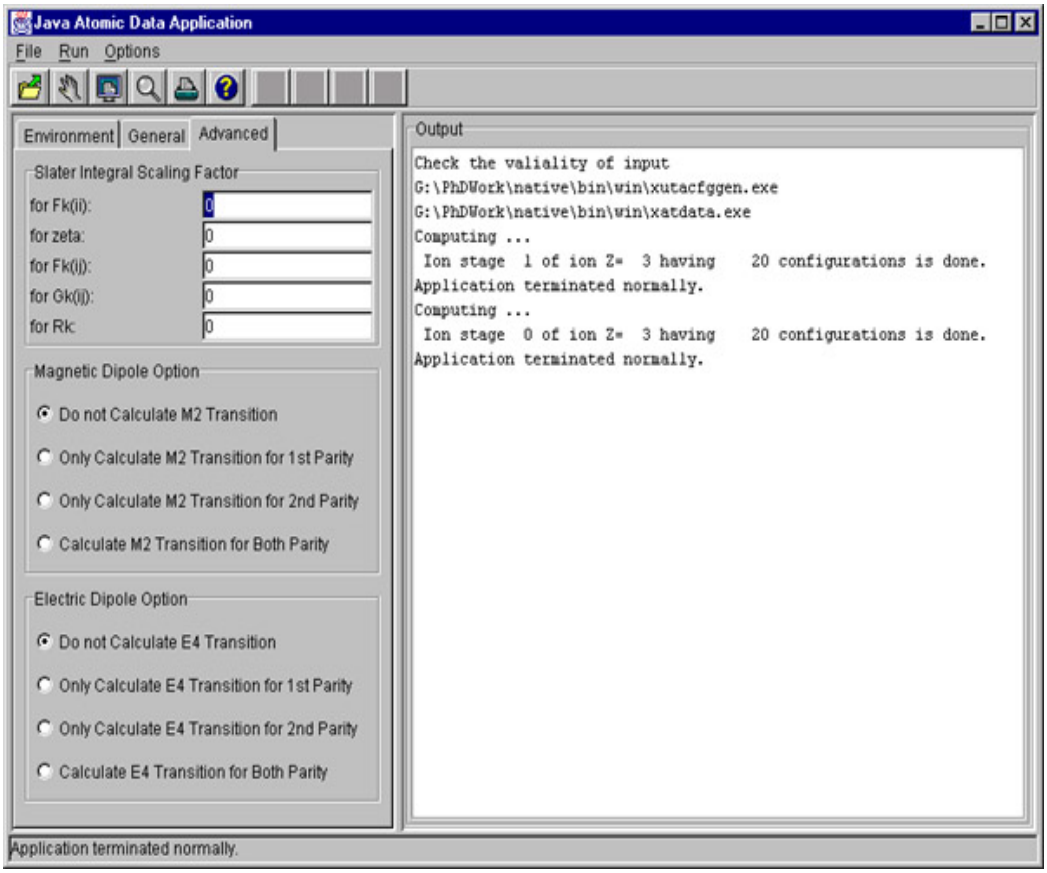


Fig. A.2.3 The advanced parameter input tab for atomic data calculation

For DTA calculations, after all raw atomic data are done, users need to specify the atomic state that will be included in the atomic table. The following panel is popped up automatically when the user chooses the DTA atomic model. N represents the principle quantum number and L represents the angular momentum of the outer orbitor. The default is the atomic table does not include the LSJ structure, includes LS structure if the outer orbitor is below 4d and includes the configuration average structure when n = 5, and above n = 5 the atomic table uses the hydrogen-like approximation.

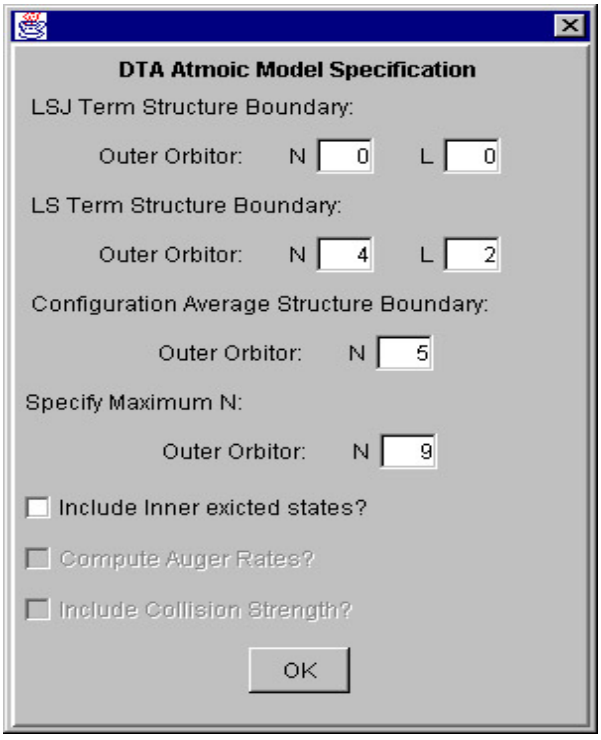


Fig. A.2.4. DTA atomic model specification

After user gives the DTA atomic model specification, the program invokes xatmodel.exe code to rearrange the raw atomic data for each ion stage. The user can see the progress from the right panel. After finishing the rearrangement, the user can construct final atomic data table by clicking the button. The following figure shows the sample results.

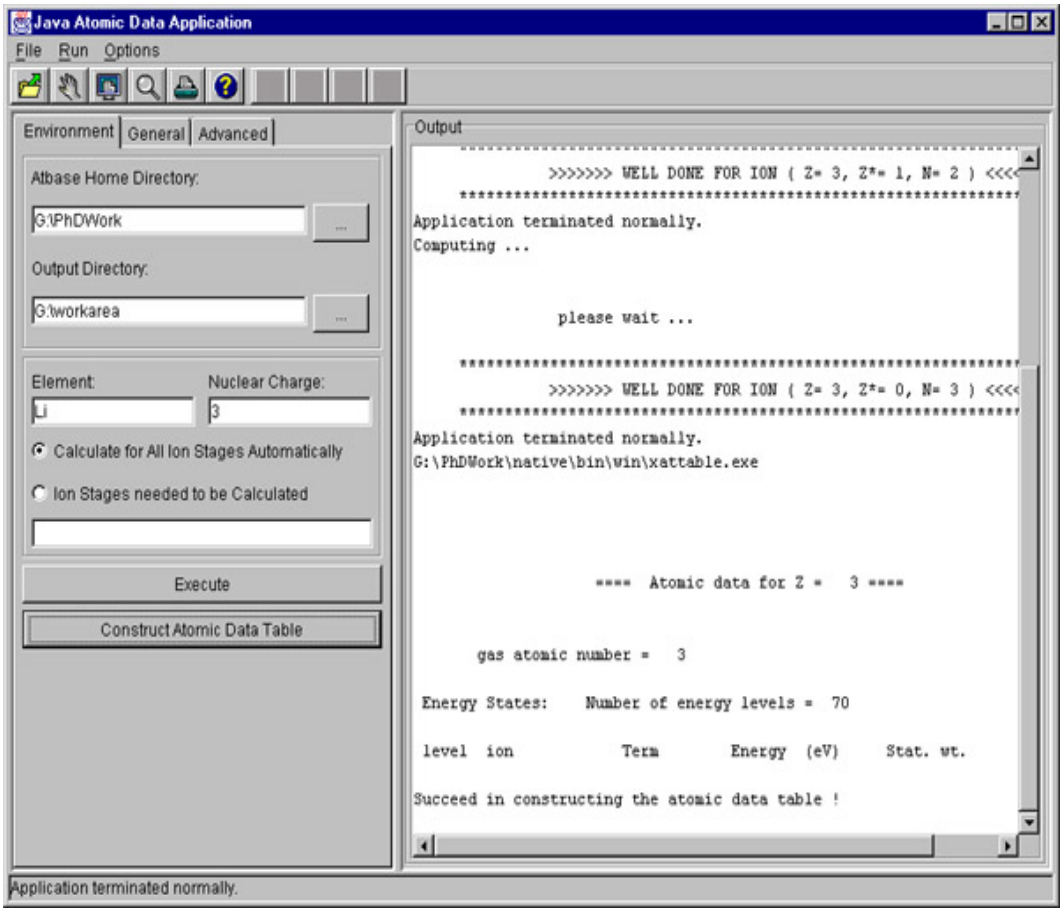


Fig. A.2.5. Atomic table generation

5. Implementation

The program is a standalone Java program. The java swing package is used extensively in the graphic user interface. To execute the native code, the program spawns another thread as shown in the following code segment:

```
class Runner extends java.lang.Thread {
    private String [] PrgArgs;
    public Runner (String [] args, Vector ions, String calcType, String outputDir) {
            PrgArgs = args;
```

```
        }
        Process runningProcess = Runtime.getRuntime().exec (PrgArgs);
        InputStream is = runningProcess.getInputStream();
        InputStream es = runningProcess.getErrorStream();
        RunningAppWatcher appWatcher = new RunningAppWatcher(is);
        RunningAppWatcher appErrWatcher = new RunningAppWatcher(es);
        appWatcher.start();
        appErrWatcher.start();
        try {
                runningProcess.waitFor();
        } catch ( InterruptedException ie ) {  }
        int exitVal = runningProcess.exitValue();
                …
}
```

6. Troubleshooting

A. Error message shows up on the right panel when executing the program.

This results from the failure of ATBASE execution. In this case, you need to redo the calculation for those ion stages that have exceptions.

III. Equation of State (EOS) and Opacity Calculation

1. Program Description

Most of the time, users doing radiation hydrodynamic calculations want a complete EOS and opacity table without knowing the details of the atomic data generation process. This program can be used in this purpose. It can calculate EOS and opacity data or mixed opacities using atomic data models DTA and UTA under the LTE approximation. Users can easily generate an EOS and opacity table or do spectrum analysis, assuming the atomic data exists. Java is used for the graphic user interface. JNI (Java native interface) and other techniques are used to communicate with the native code. The underlying native code is based on EOSOPA (see UWFDM-933, Ping Wang, 1993).

2. Executing Environment

Sun JDK1.2 or later versions are required. Microsoft Visual Studio 5.0 is used to compile and execute the FORTRAN code. The native programs include

1. xdtaopa.exe: Calculate LTE and non-LTE EOS and opacity under the DTA atomic model.
2. xutaopa.exe: Calculate LTE EOS and opacity under the UTA atomic model.

The file structure is shown in Appendix B. This program has been tested successfully on Windows NT.

3. Program Outline

The EOS and opacity calculation module provides a friendly user interface to generate an EOS and opacity table for hydrodynamic calculations and for the spectrum analysis under the LTE approximation. Users specify the Atbase home directory so that the program can locate the executable programs and other parameters required. The home directory must be correct since the program detects all available atomic data automatically according to the home directory. The output directory assigns which directory the output table will go. Only the LTE plasma model is supported currently. For the atomic data model, users can choose the DTA model or UTA model. In the element list box, all available atomic data under the user's chosen atomic model will be listed. The fraction list box gives the fraction of this element in the compound. After clicking the Add button, the element name and the fraction will appear in the plasma constitution list box.

Users can also change the temperature grid, the density grid and the photon energy grid using the *T Mesh*, *D Mesh* and *P Mesh* tabs, respectively. It allows user to specify the minimum value and the maximum value and the mesh number. Users can see the grid in the table in either linear scale or log scale. There is an option that lets the user decide whether to include Stark broadening. For UTA calculations, usually the Stark broadening should be included. However, for the spectrum analysis in the DTA model, users are more interested in the individual line and the UTA width should not be

included. The program automatically checks this option according to the atomic data model.

The program executes the native code after users click the *Execute* button. First, the validity of input is checked. Then, the corresponding atomic data are copied from the Atbase home directory to the work directory. The native code is then executed and the debug output information will show in the right panel. The *View Results* button will be enabled after the calculation is finished. By clicking this button, it brings up a menu which gives the options that display the results. If viewing the table as a graph is chosen, the program will invoke the OpViewer module (see *EOS and Opacity Viewer*). If users want to see the orbital occupation probability or frequency dependent opacity or absorption spectrum, they must choose one temperature point and one density point, otherwise, these options will be disabled.

The *Orbital Occupation Probability* shows the occupation probability for each orbitor graphically. The *Frequency Dependent Opacity* displays total opacity and three components (bound-bound, bound-free and free-free). The user can manipulate the graph by right-click to pop up a menu (see *EOS and Opacity Viewer*). The *Absorption Spectrum* gives the absorption spectrum based on the optical thin model. Users can adjust the optical depth. The file names of the output EOS and Opacity tables are utaopa.TAB for the UTA model and dtaopa.TAB for the DTA model.

4. User Interface

In the following, some screen shots for the EOS and opacity calculations under both DTA and UTA model are given. Because they are sample screen shots, no detailed physics implication has been discussed.

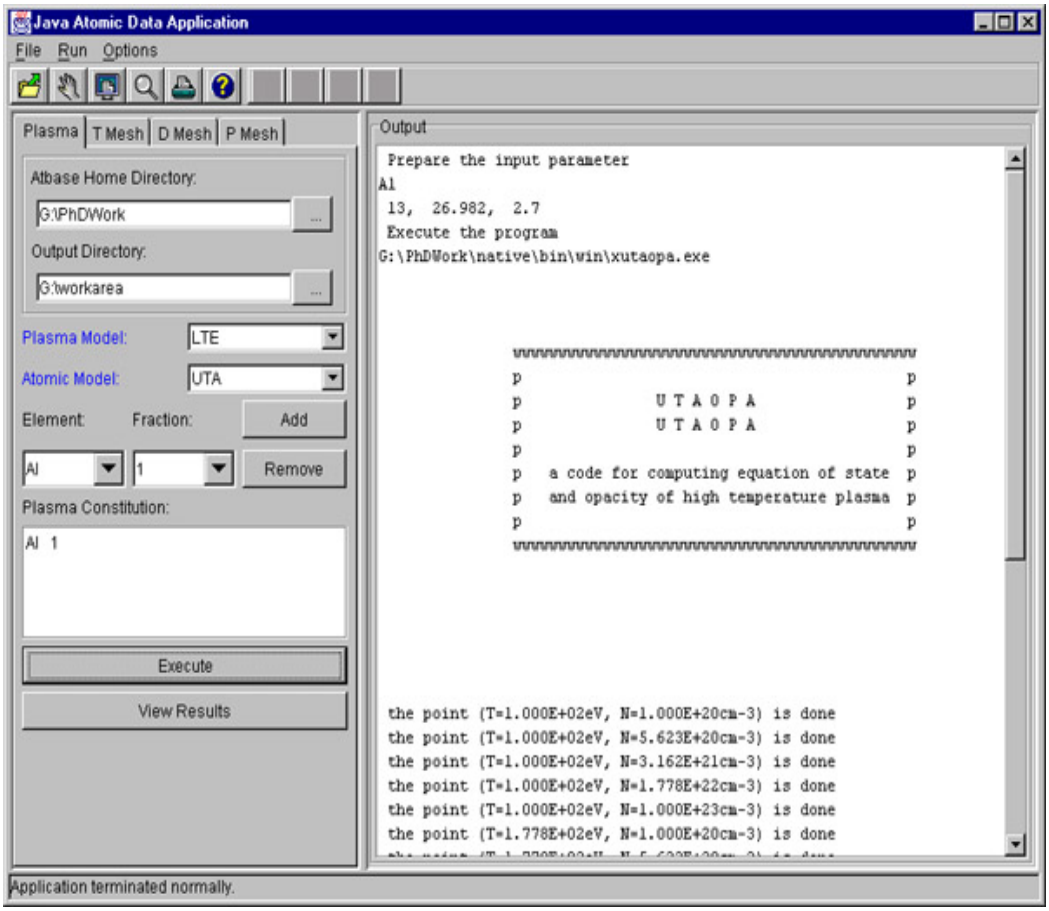


Fig. A.3.1. The plasma condition tab in EOS and opacity calculation

In this tab, users give the Atbase home and output directory by clicking the directory browsing buttons. Currently, only the LTE plasma model can be used. For the atomic model, the user can choose to use the UTA model or the DTA model. The

available atomic data that will be used in the EOS and opacity calculations will be automatically searched under the *Atbase home directory/native/data/DTAdata or UTAdata.* The DTA atomic data files for each element are atomic.dat and pixfit.dat in its own directory, while the UTA atomic data files are uta.atomicdata and uta.photonizxx in its directory. The corresponding data will be copied into the work directory when issuing the *Execute* command according to the atomic model used. The specification of a compound is given by using the *Add* and *Remove* button.

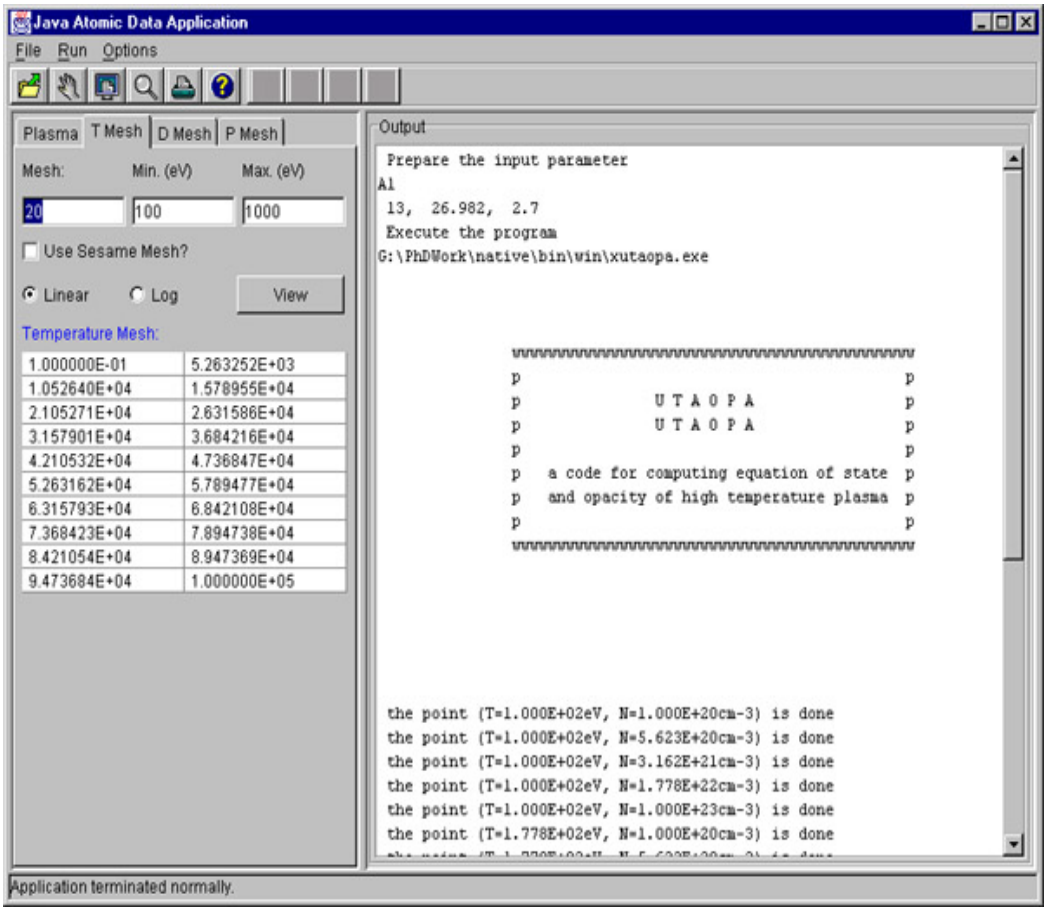


Fig. A.3.2. The temperature mesh tab in EOS and opacity calculation

In the temperature mesh tab, the user specifies the temperature mesh for the EOS and opacity table. There are two mesh types: linear and log. The user can see the mesh by clicking the *View* button. If the Sesame mesh is used, the first mesh point will be the first one that is greater than the minimum value and the last one will be the one that is smaller than the maximum value. If using only one temperature point, the temperature is the minimum value.

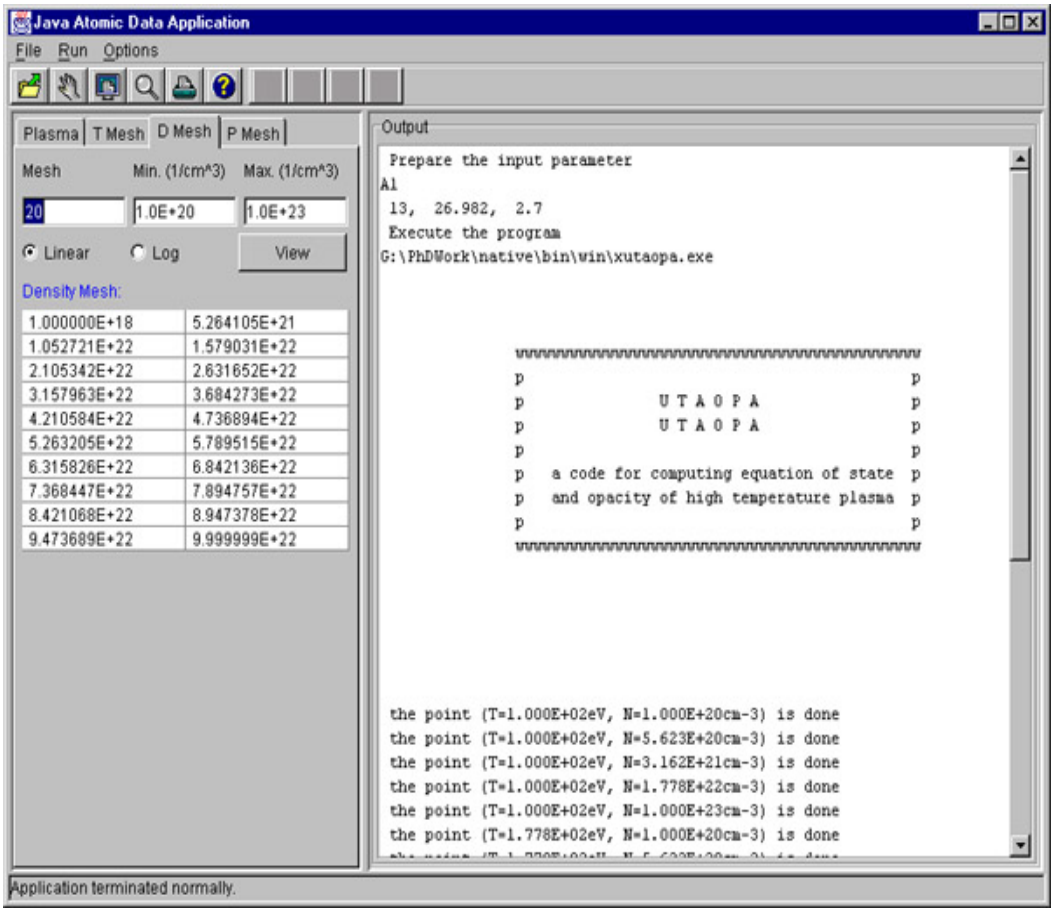


Fig. A.3.3. The density mesh tab in EOS and opacity calculation

In the density mesh tab, the user specifies the density mesh for the EOS and opacity table. There are two mesh types: linear and log. User can see the mesh by clicking the *View* button. If using only one density point, the density is the minimum value.

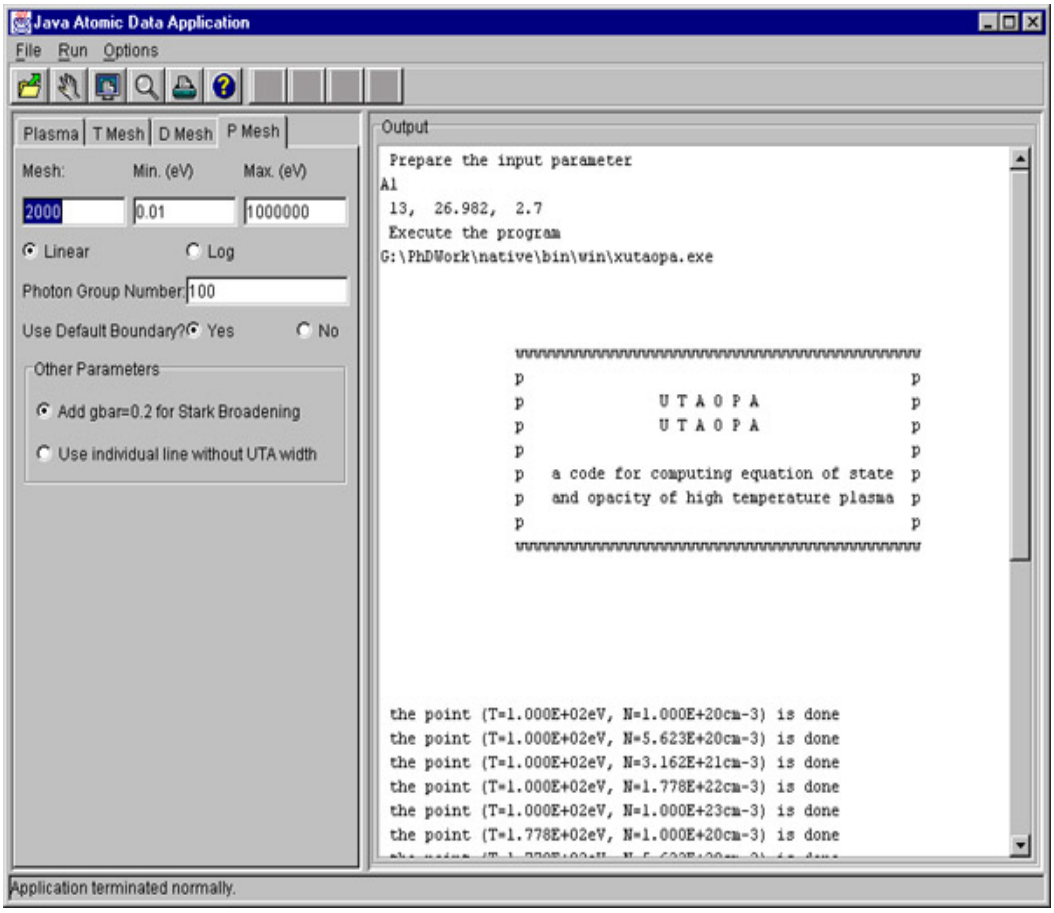


Fig. A.3.4. The photon energy mesh tab in EOS and opacity calculation

In the photon energy mesh tab, the user specifies the group photon energy boundary mesh for the opacity table. There are two mesh types: linear and log. To calculate the total opacity quantities, the user can let min=0.01eV, max=1000000eV and use 1 photon group. Otherwise, given the photon group number and the minimum and maximum photon energy values, the program can generate an opacity table. Users can choose to use their own photon boundaries. After clicking the *No* radio button, a form appears to allow users to either input the photon boundaries in the text boxes or

import a file that specifies the photon boundaries. To analyze the spectrum, the user should give more mesh points in the spectrum range in order to have a better resolution. For the UTA calculations, the Stark broadening mechanism is included. For the DTA calculations, the default is to use the individual lines.

After the calculation, a menu to display the results is shown. If the user calculates an EOS and opacity table (T >1 and D>1), the detailed information button will be disabled as shown in Fig. A.3.5. The user can only use *As Graph* button to visualize the results. However, if the user calculates opacity or spectrum for one temperature and one density, those buttons will be enabled as shown in Fig. A.3.6.

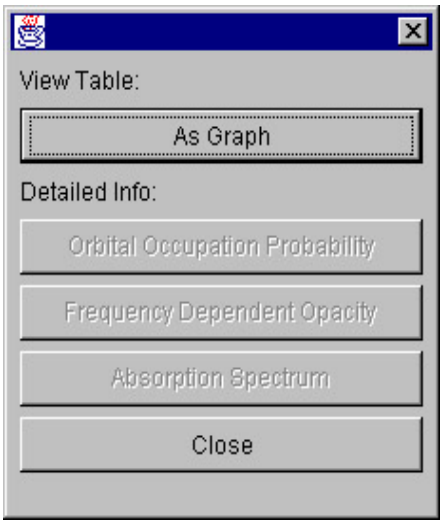


Fig. A.3.5. The menu for displaying results after generating an EOS and Opacity table

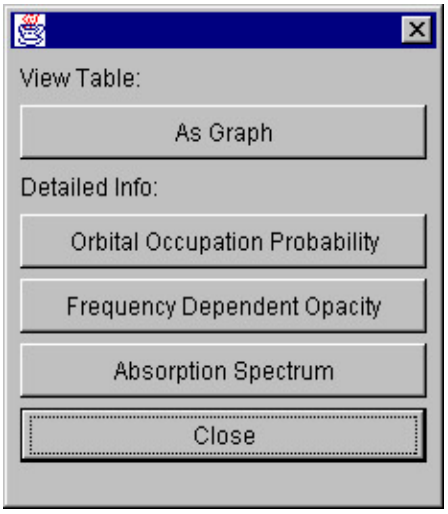


Fig. A.3.6. The menu for displaying results for a single temperature/density pair.

The following sample figures show the orbital occupation probabilities for Al plasma and a mixture of Al, C, H and B.

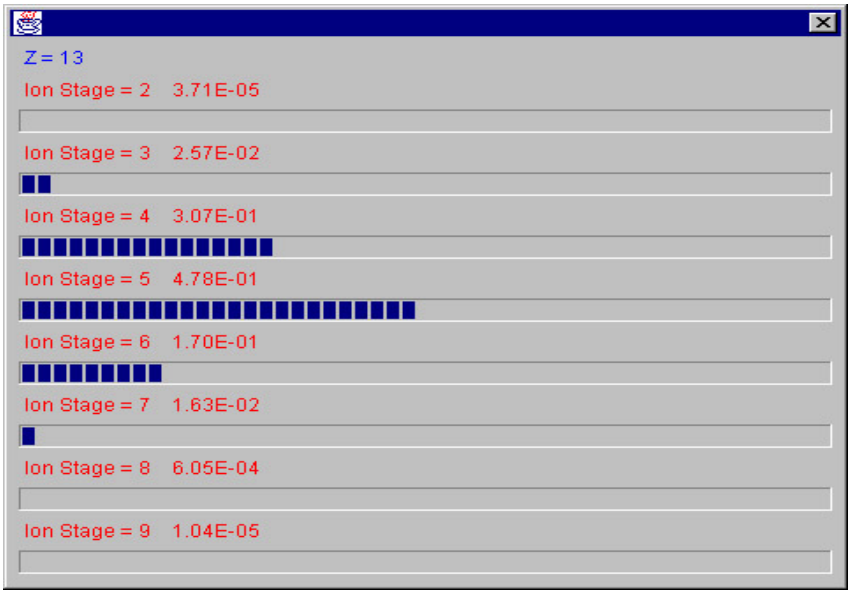


Fig. A.3.7. The orbital occupation probability for Al plasma.

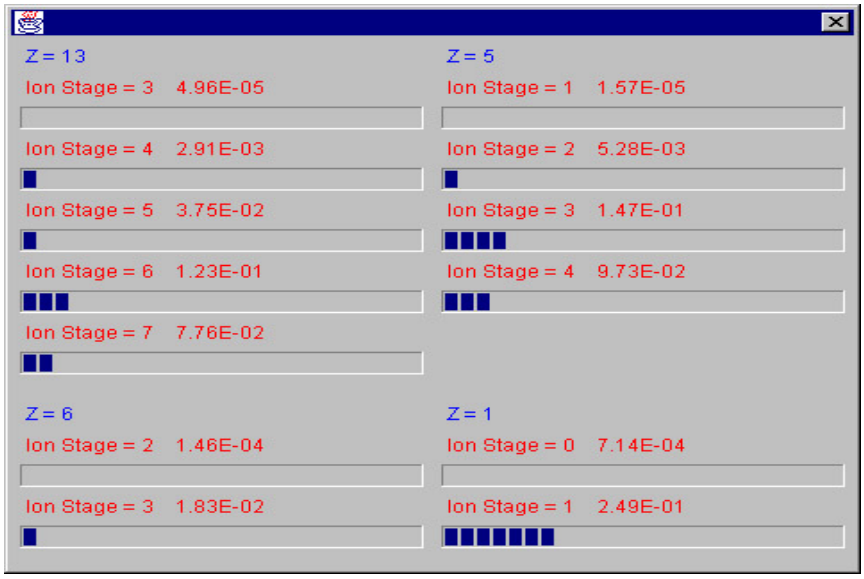


Fig. A.3.8. The orbital occupation probability for the mixture of Al, C, B and H

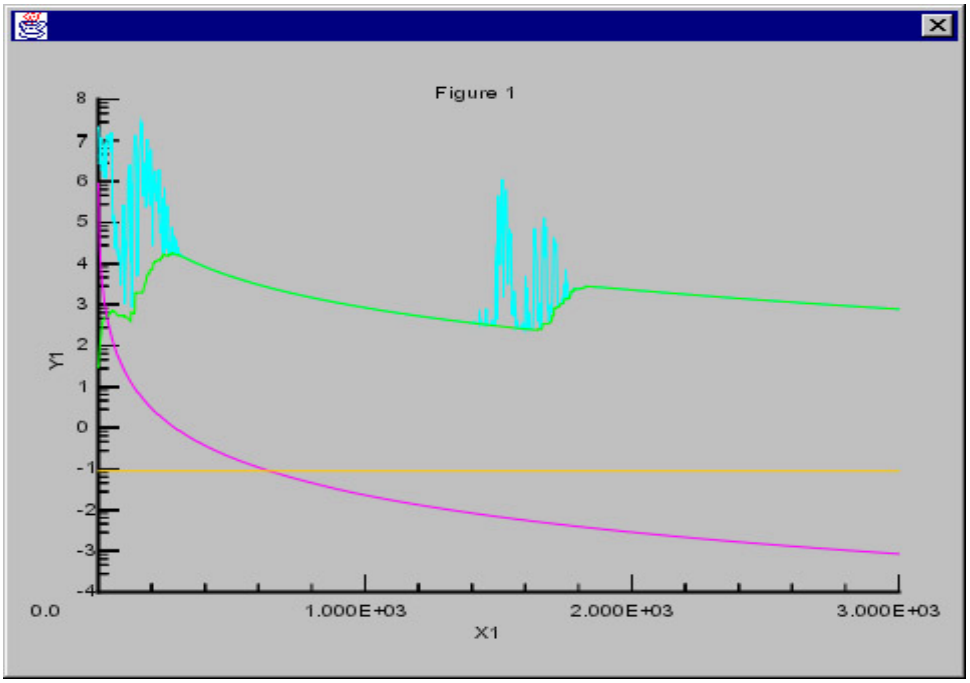


Fig. A.3.9. The three components (bound-free, free-free and scattering) that contribute to the opacity.

The above figure shows the three components of the opacity and the total opacity in the log scale (bound-bound transition can not be shown in the log scale) using the DTA model.

The user can see the bound-bound contribution to the total opacity in the linear scale, as shown in Fig. A.3.10. How to manipulate the graph is discussed in Appendix A. IV.

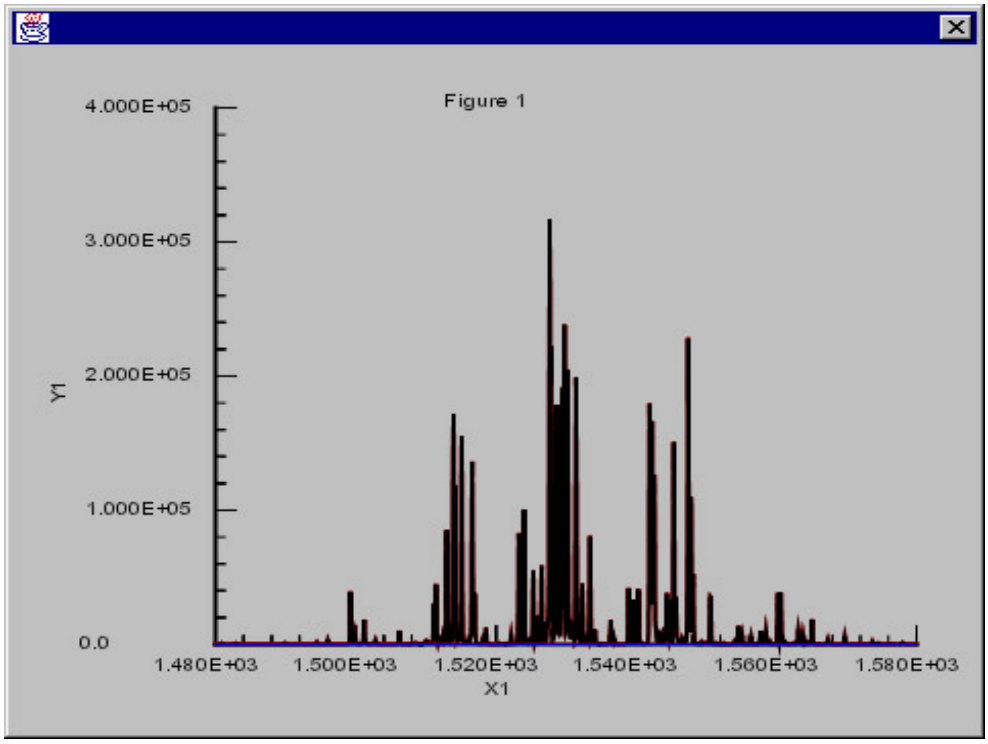


Fig. A.3.10. The four components of opacity in the linear scale.

The user can also obtain an LTE optical thin absorption spectrum. By adjusting the plasma length, user can easily see the effect.

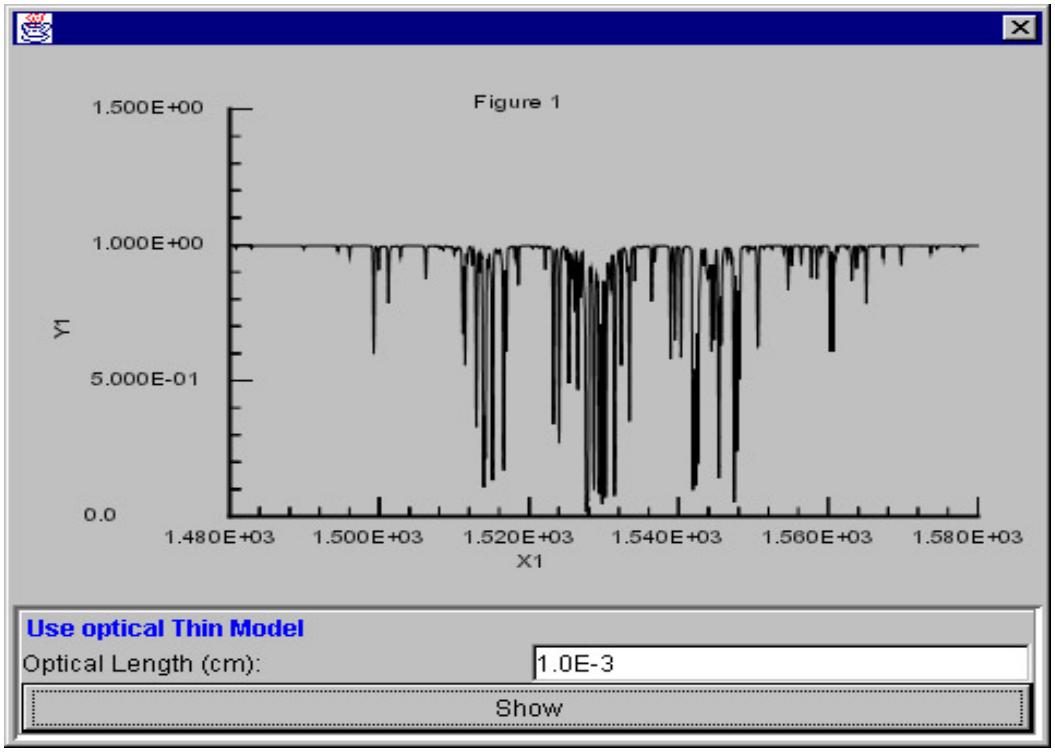


Fig. A.3.11. The LTE absorption spectrum using the optical thin model.

5. Implementation

The program is basically a standalone Java program. The java swing package and java thread technology are used extensively. The program spawns a new thread whenever it needs to do an execution in order not to freeze the interface (see the example in *Atomic Data Calculation*). Native *copy* functions are used whenever the program needs to copy files from one directory to another directory for better performance. To display the EOS and opacity results, the package *JAtbase.AtGraph2d* is used.

6. Troubleshooting

A. Error messages like " error when calculating the heat capacity " occur.

This results from the failure execution of the UTAOPA or DTAOPA code. These codes may have problems (underflow or other exceptions) when calculating certain temperature and density points. In this case, you can adjust the temperature or density grid a little, hoping it can skip those bad points.

IV. EOS and Opacity Viewer

1. Program Description

This module is used to visualize the EOS and opacity results. It can run both as a standalone Java application and as a Java applet in a web browser. The data source can be from the local file system for Java application or from a remote data source for both Java application and Java applet. The user has options to view each part of the EOS and opacity. Each panel on the right side concurrently displays the results and a pop up menu can help manipulate the graph. The user interface and the graph package are programmed using JDK1.2 API. For the remote data source, several JavaServlets residing in the application server serve as the remote data provider.

2. Executing Environment

Sun JDK1.2 or above are required to run this Java application. To run the program in a web browser (Netscape or Internet Explorer), the Java 1.2 plug-in is needed.

3. Program Outline

The EOS and opacity viewer module provides a flexible tool for users to visualize the EOS and opacity results. The data format that is known by this program is the BUCKY EOS and opacity data format. Users can load the data from the local machine by browsing the directory and specifying the data file (usually utaopa.TAB or dtaopa.TAB). It takes a while for the program to load the data (usually about 3M bytes) and split it up into smaller parts for the graph display. If user chooses to load data from

the remote data source (that is, capsule.neep.wisc.edu), the program first contacts a JavaServlet residing in the application server and obtains all available elements in the database. The data source panel will show a list box and all available elements will appear in that list box.

The program has options that let the user see each part of the EOS and opacity table. If *View All Data* check box is checked, all EOS and opacity data will be automatically drawn on the right tabbed panels. The user can select some parts to draw by disselecting the *View All Data* check box and choose from the check list box on the *Opacity* and *EOS* tabs.

After the *View* button is clicked, the program decides the number of tabbed panels that need to display the data, then each panel spawns a separate thread to load the data from the data source and render the data. The temperature, density and photon energy grids are also shown on the panels for reference. The graphs of EOS data will be automatically drawn on each panel. For opacity data, the user needs to click the upper bar on the panel to bring up a temperature and density grid panel, which is used to determine the opacity data set that will be drawn.

The graphs that users display may be not good because of the large scale of the data range. Users may manipulate the graph through a graph controller by right-clicking within the graph frame. In the graph controller, the number of data sets is listed and other graph controls such as the line size, color, data range, axis labels are shown.

Users can use it to obtain a much better graph. The *info* button tells the size of the data set and the kind of data. Users can also print the graph from the popup menu.

4. User Interface

In the following, some screen shots are given. Because they are sample screen shots, no detailed physics implication has been discussed.

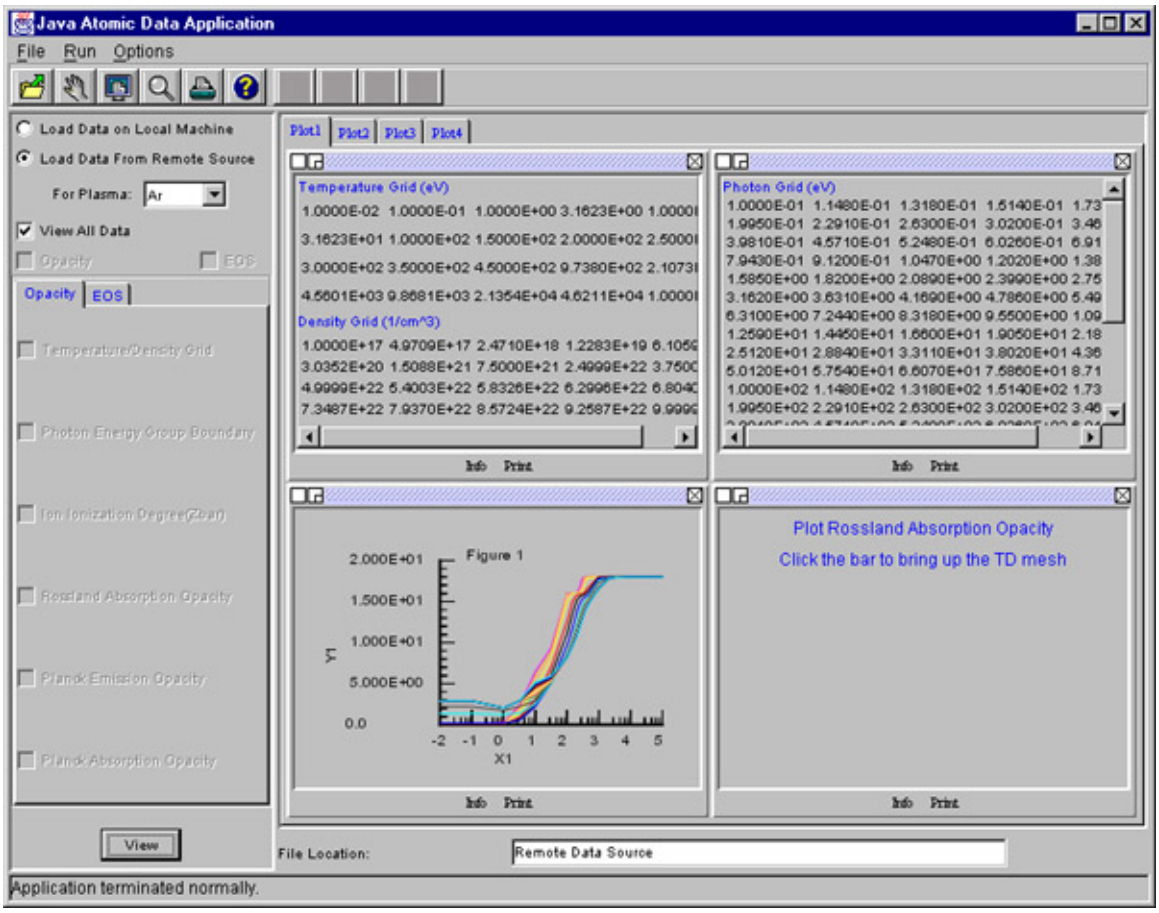


Fig. A.4.1. Load data from remote source and display all results automatically

The program determines the number of tabbed panels required and draws the data concurrently in each panel.

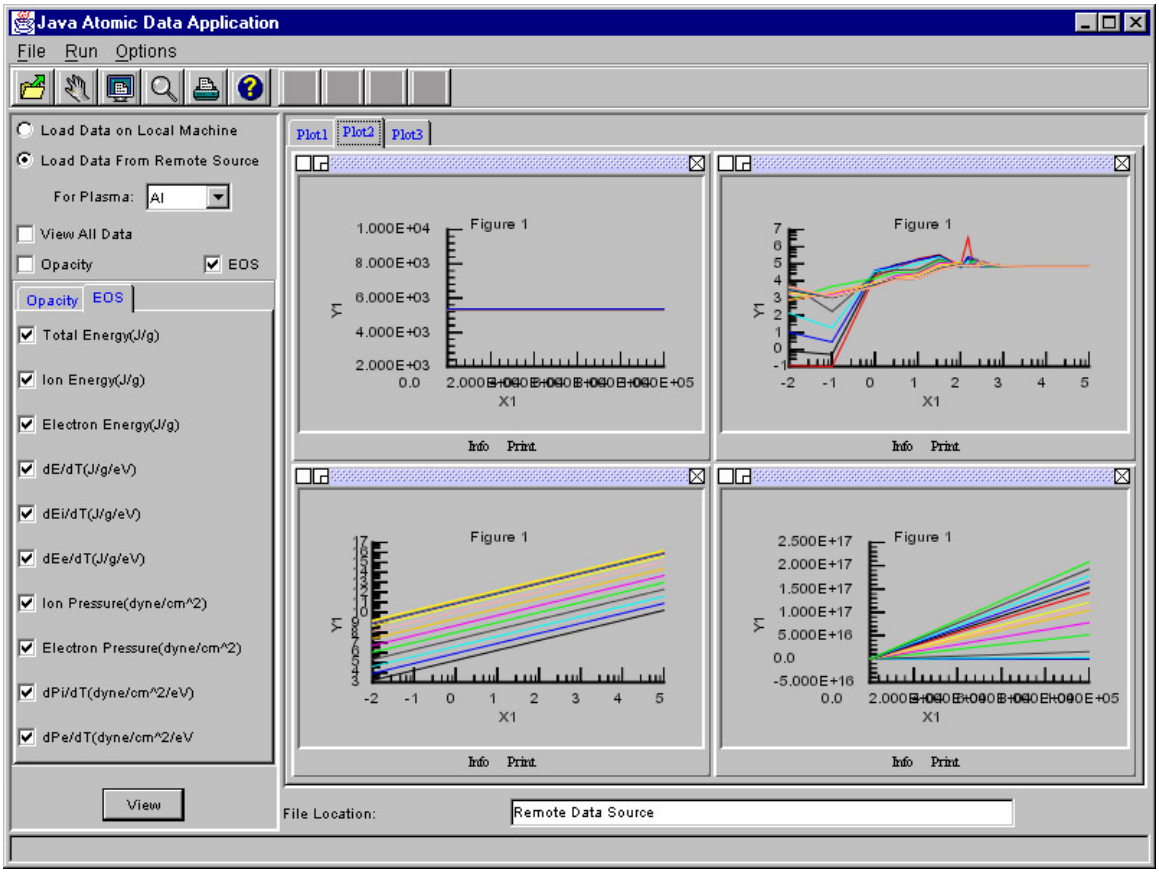


Fig. A.4.2. Display selected EOS results.

Users can select the components of the EOS and opacity. The program determines the number of tabbed panels required and draws the data in each panel.

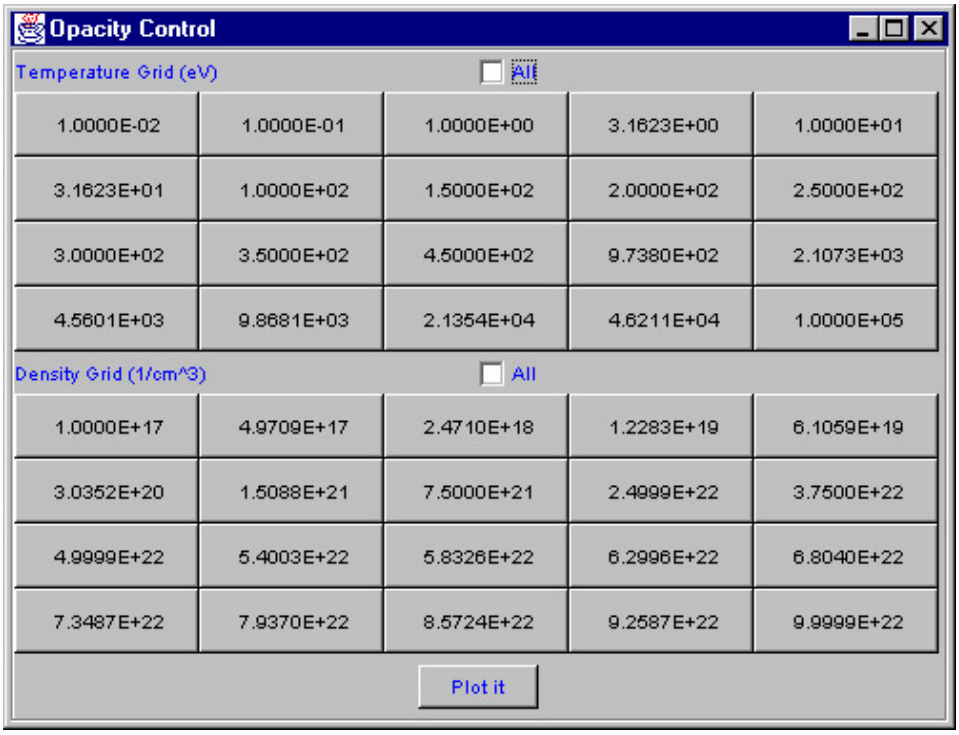


Fig. A.4.3. Temperature and density grid

Users can draw opacities in T-P (temperature-photon energy) grid or D-P (density-photon energy) grid easily by selecting the points shown on the above panel. Only one temperature point or one density point is allowed, otherwise, there are warning messages.

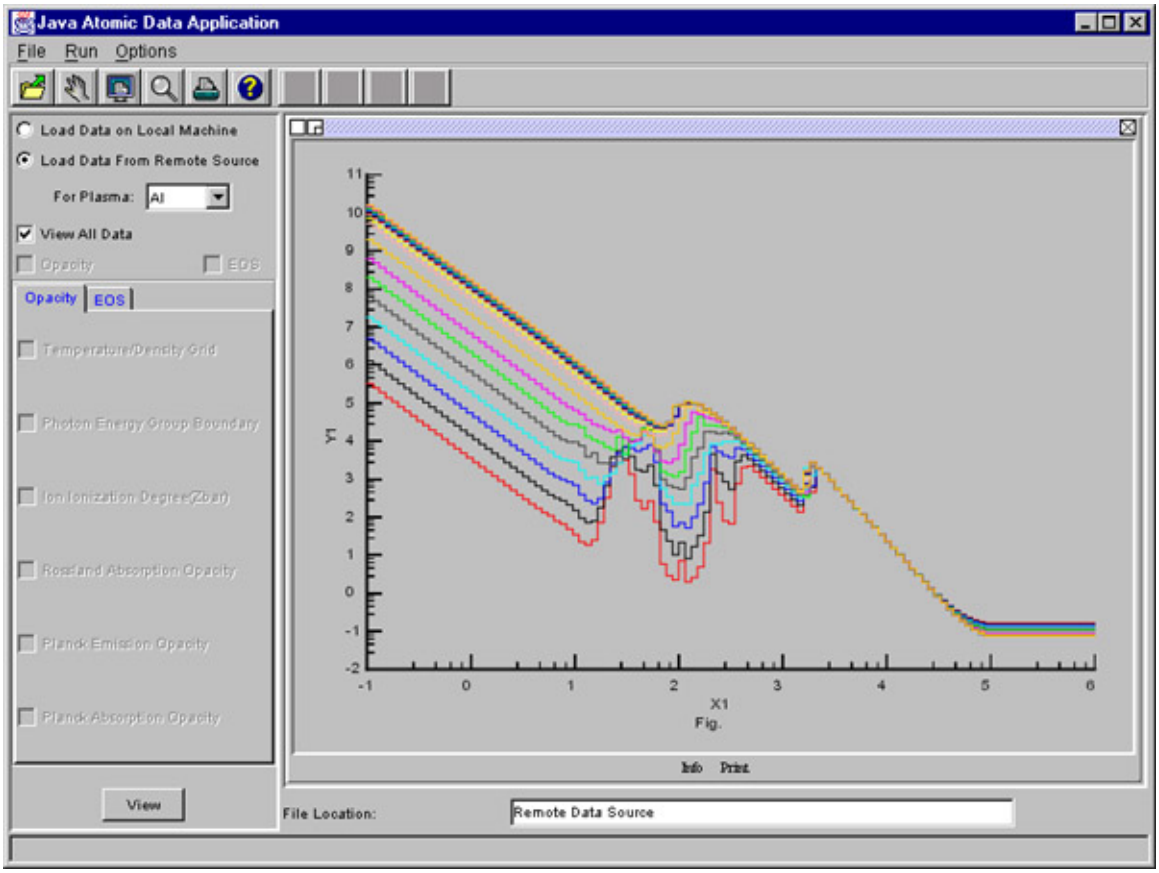


Fig. A.4.4. Opacity data display

Figure A.4.4 shows a sample of Rossland 100 group opacities. It is maximized by clicking the small icon on the upper bar. To restore this graph to its original position, the restore icon should be clicked. This graph has been manipulated by the graph controller.

The graph controller shows information in two categories: data set and axis. In the data set section, it shows the data information. To disable or enable some data sets, the user can select the data sets from the left list panel and click the *Enable* button or the *Disable* button. To change color or the line pattern or the line size for a curve, the

user need to first select the data set from the left list panel. Otherwise, there is a warning.

Because the same GUIs are used for X and Y axis, some fixed orders of operation are required. In the following, it shows how to change the data range, set axis labels, draw major and minor grid and set the graph title.

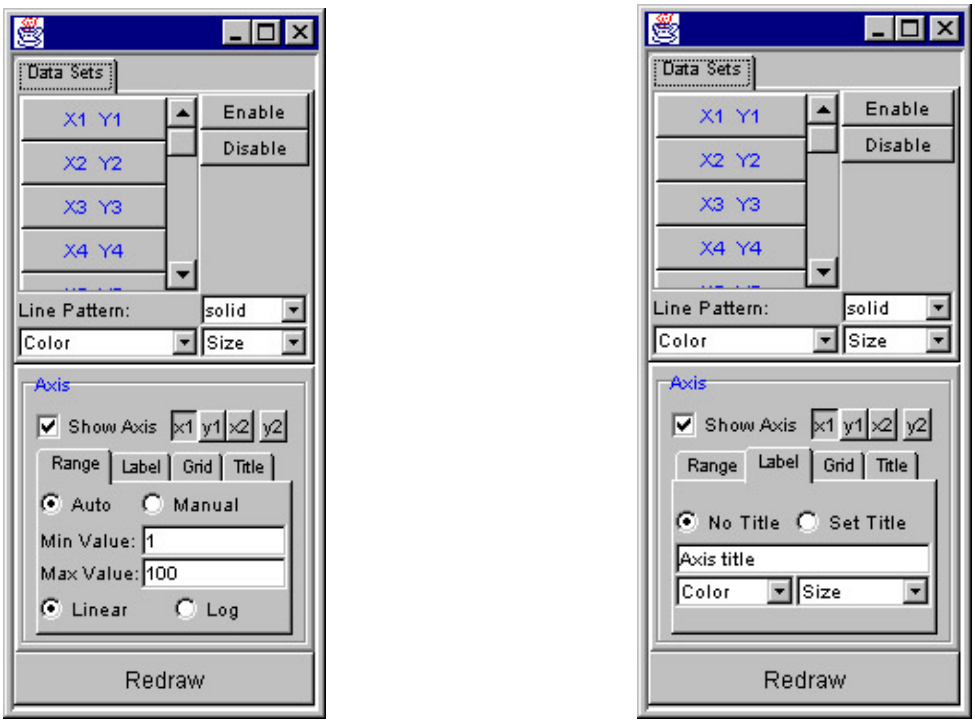


Fig. A.4.5 Set up the data range and the axis label

To set the data range to be drawn (see Fig. A.4.5), the user should click the button in the order of *x1* or *y1*, *Auto* or *Manual*, *Min* or *Max Value*, *Linear* or *Log*, and then *Redraw*. Make sure the x1 button is chosen when manipulating with the x axis and the y1 button is chosen when manipulating with the y axis.

To set the axis label (see Fig. A.4.5), first make sure the axis name button is chosen, then type the axis title in the text field box and choose the color or the font size for the axis title, click the *Set Title* radio button next, and finally click *Redraw* button.

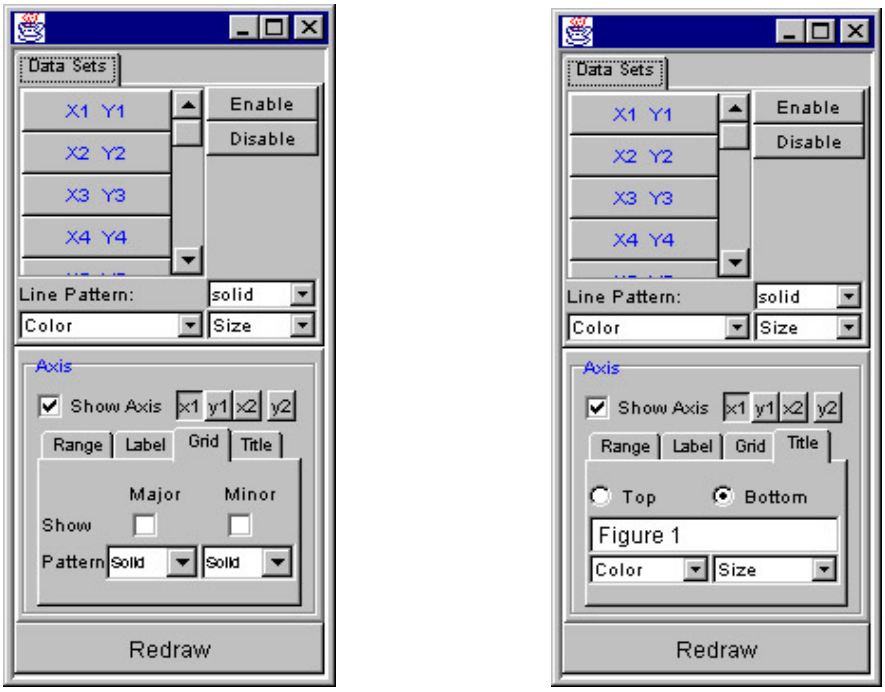


Fig. A.4.6 Draw the major and minor grid and set the graph title.

To add the major grid or the minor grid to the graph, check the *Show* box and choose the line pattern and then click the *Redraw* button (see Fig. A.4.6).

To set the graph title, first type the graph title in the text filed box and choose the color and the font size for the graph title, and then choose the graph title position by clicking the Top or Bottom radio button, finally click the *Redraw* button (see Fig. A.4.6).

5. Implementation

The program is designed to run both as a standalone Java program and as a Java applet. The Java swing package is used for the graphic user interface. Java threads are used extensively for better drawing performance. A package named *JAtbase.AtGraph2D* is designed using Java2D, which can be used as a graphic tool for general purpose. This package provides the external interface that lets the user plug in his own graph control object. The implementation of the mouse listener can handle the mouse operation.

For the remote data source, several JavaServlets reside on the application server (capsule.neep.wisc.edu), acting as the data provider. These Java Servlets then contact with the data server (lapop.ep.wisc.edu) to retrieve the data from the database.

For the local data source, the Java program directly retrieves the data by reading the data file. A common data loading interface is designed to unify the data loading operation from the local data source and remote data source.

6. Troubleshooting

None.

IV. Spectrum Analysis For High Z Plasmas

1. Program Description

This module can be used to perform spectrum analysis for laser-produced high Z plasmas. The atomic data are calculated using the RSSUTA model on NPACI IBM SP machines. The data are then imported into the Oracle database. CORBA framework paves a way to connect the database and the graphic user interface. Currently, only the LTE plasma model is supported.

2. Executing Environment

This module is based on the commodity three-tier architecture. Oracle 8I database management system and Oracle Application Server 4.8 are used. Sun JDK1.2 or above are required to run Java application.

3. Program Outline

After the atomic data have been calculated using the parallel code RSSUTA, they are transferred to the local machine and then imported into the Oracle database by running a script. The program first tests the connection between the database and the graphic user interface. If it succeeds, it will list all available data in the database in the list box. After specifying the temperature and density points and the photon energy range, users can invoke the calculation by simply clicking the *Show* button.

The program loads the first lowest ten relativistic configurations and solves the Saha equation to guess the most important ion stages, then it retrieves all average

configuration energies for each of the important ion stages and caches these data for later use. The Saha equation is solved to obtain the population of these configurations. There are generally thousands of configurations involved. Minimum of 1.0E-3 is used to determine the configurations that will be used in the spectrum generation and to filter those configurations with too small populations. The program then retrieves the photoexcitaion and photoionzation cross sections for each of these configurations. The data required by the UTA spectrum analysis such as energy positions, transition oscillator strength and width are obtained from the database. With configuration populations and the spectrum data, the program finally constructs the spectrum and shows the result on the right panel.

Users can get detailed information by clicking the *Data Information* button. The ion stage distribution and the calculated ionization energies are displayed. Users can see the relativistic electron configurations involved in the calculation for each ion stage and the transition data and ionization data for each configuration.

4. User Interface

The following will show some screen shots as an example in the spectrum analysis for Ge plasma (T =76eV and D = $2 \times 10^{20} cm^{-3}$). We will only focus on the graphic user interface instead of physics.

Fig. A.5.1 provides general information of this run such as plasma condition, average ionization and ion stage distribution. The calculated ionization energies for this element can be obtained by clicking the *Ioniz Energy* button, as shown in Fig. A.5.2. Users can see how many configurations are involved for an ion by choosing from the ion distribution table and clicking the *Configurations* button.

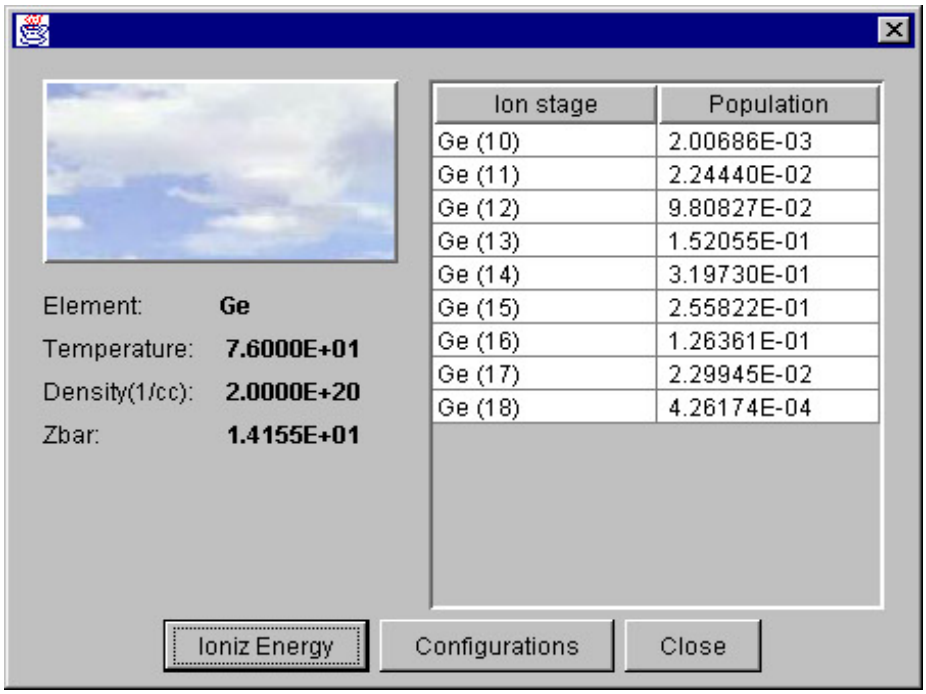
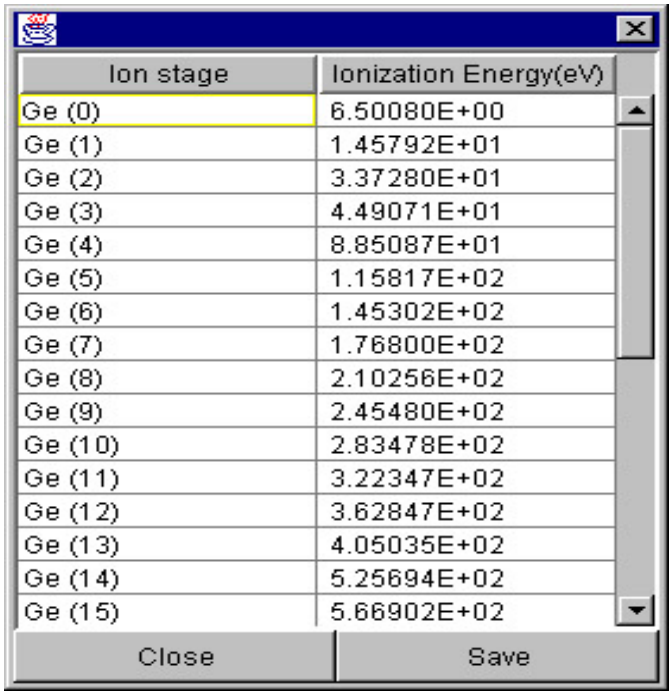


Fig. A.5.1 General information of plasma condition and ion distribution



Ion stage	Ionization Energy(eV)
Ge (0)	6.50080E+00
Ge (1)	1.45792E+01
Ge (2)	3.37280E+01
Ge (3)	4.49071E+01
Ge (4)	8.85087E+01
Ge (5)	1.15817E+02
Ge (6)	1.45302E+02
Ge (7)	1.76800E+02
Ge (8)	2.10256E+02
Ge (9)	2.45480E+02
Ge (10)	2.83478E+02
Ge (11)	3.22347E+02
Ge (12)	3.62847E+02
Ge (13)	4.05035E+02
Ge (14)	5.25694E+02
Ge (15)	5.66902E+02

Close Save

Fig. A.5.2 The calculated ionization energies

In Fig. A.5.3, we show the relativistic electron configurations for P-like Ge that have been included in the calculation. The population for each configuration is also shown in this figure. For each configuration in this table, users can obtain more detailed information about transition arrays and ionization edges by click the *Transition Lines* button and the *Photonization Edges* button. Samples are shown in Fig. A.5.4 and Fig. A.5.5.

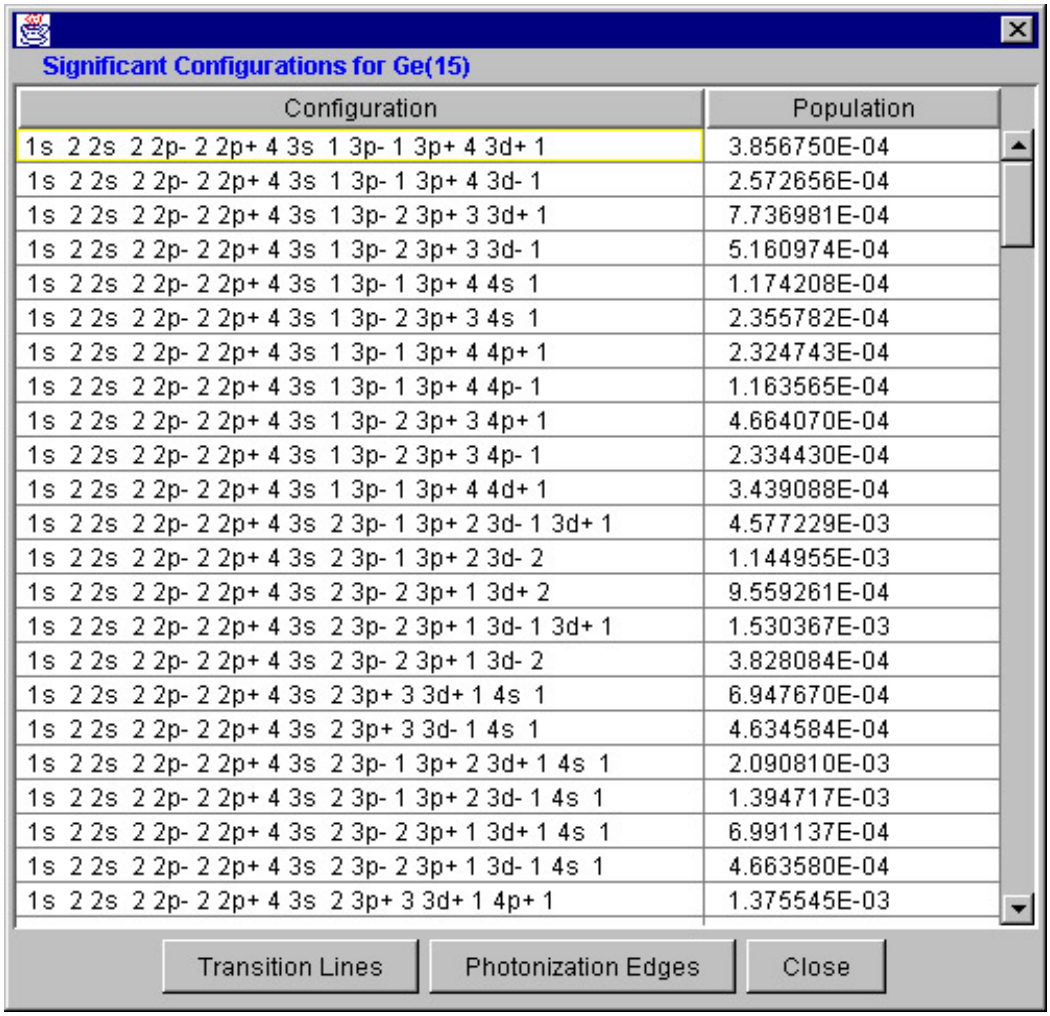


Fig. A.5.3 Display of relativistic electron configurations that are involved in the calculation for P-like Ge ion. The configuration populations are also shown.

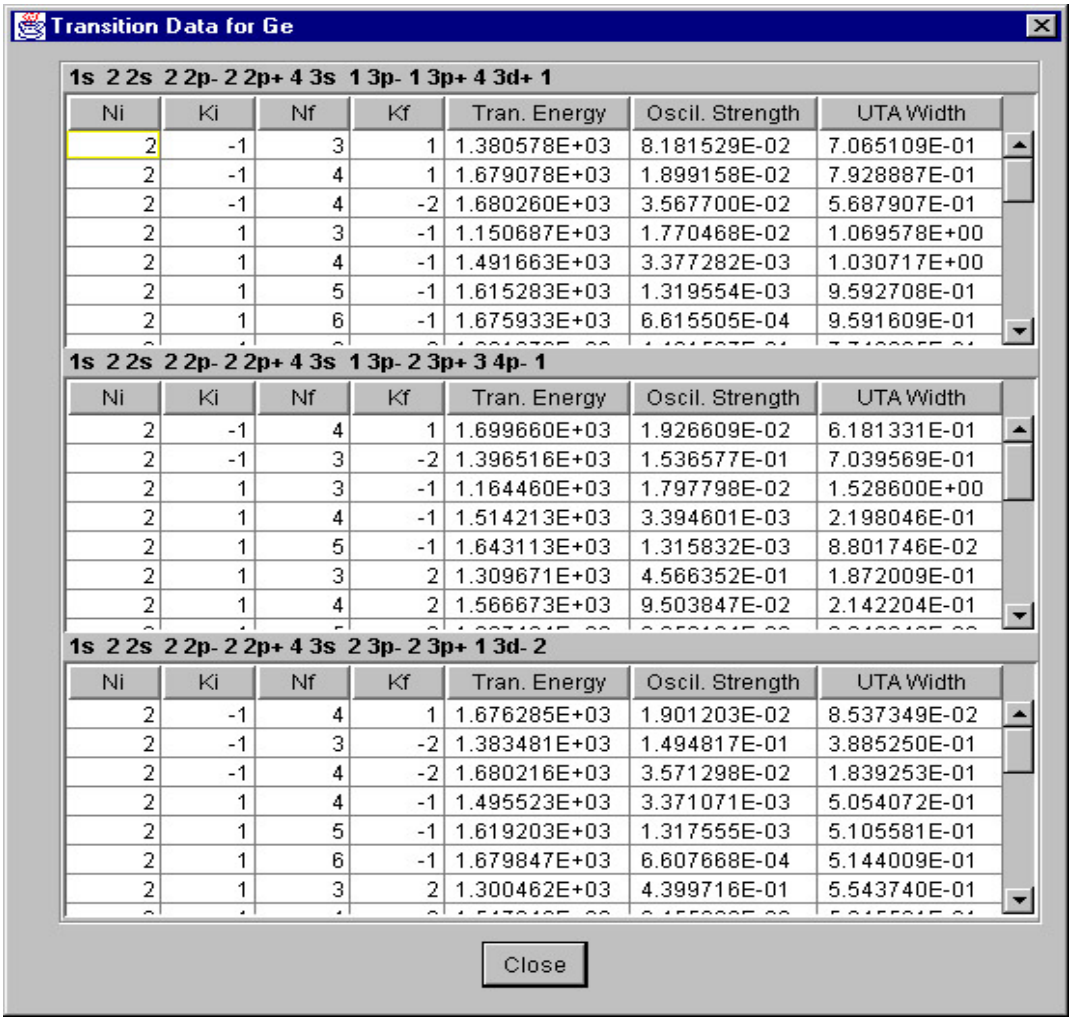


Fig. A.5.4 Detailed transition data for three configurations. The configuration is shown as a label above the subtable. The initial and final orbitals, transition energy, oscillator strength and UTA width are also given.

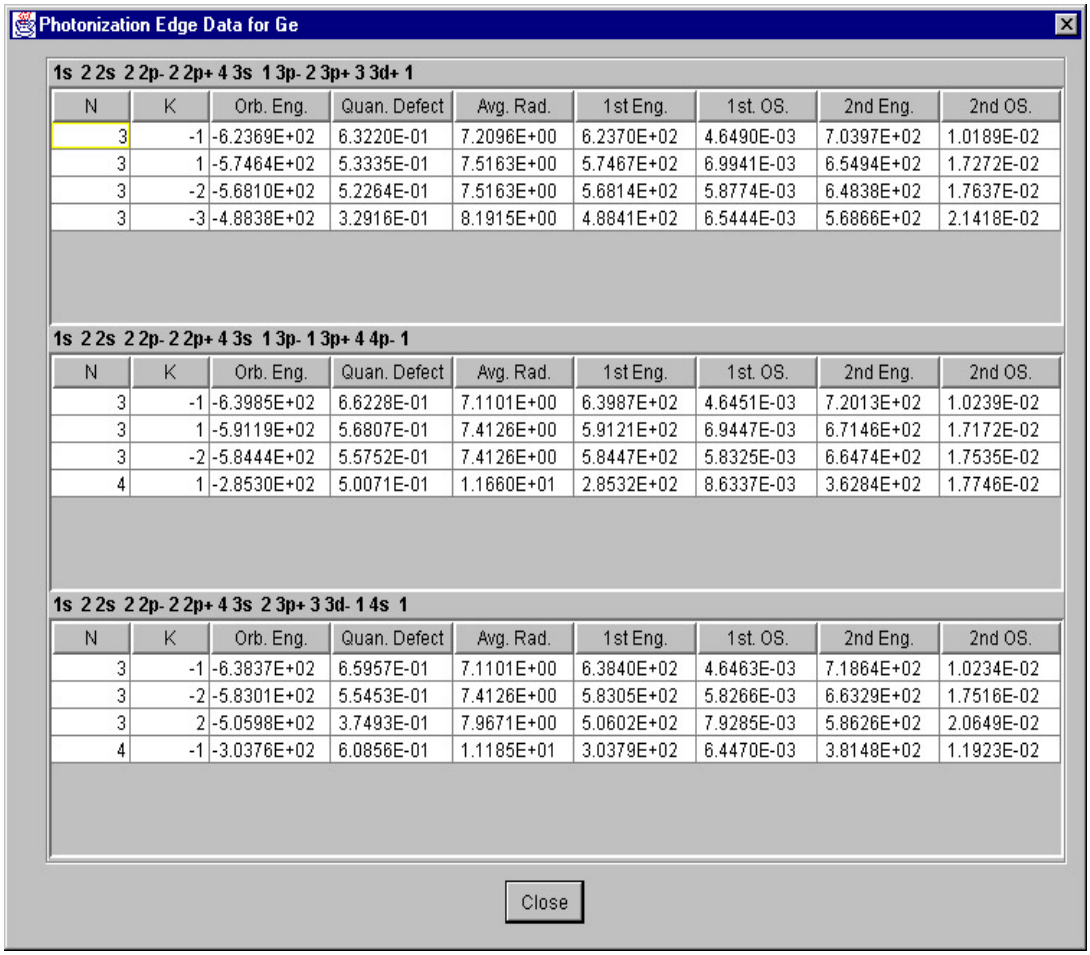


Fig. A.5.5. Ionization edges for three configurations. A pair of N and K represents the electron orbital. Label *Orb. Eng.* represents the orbital binding energy and therefore the ionization edge. Label *Quan. Defect* represents the quantum defect of this orbital. Label Avg. Rad. represents the average radius <r> of this orbital. Labels 1st Eng., 1st.OS, 2nd Eng., 2nd OS. represent the first two points of the photoionzation oscillator strength.

Users can also visualize the contributions from the bound-bound transitions or the bound-free transitions or from individual ion stages. The plasma path length is also shown. Note these contributions are calculated using this path length. Emission spectrum can also be constructed using the same data retrieved from the database.

.

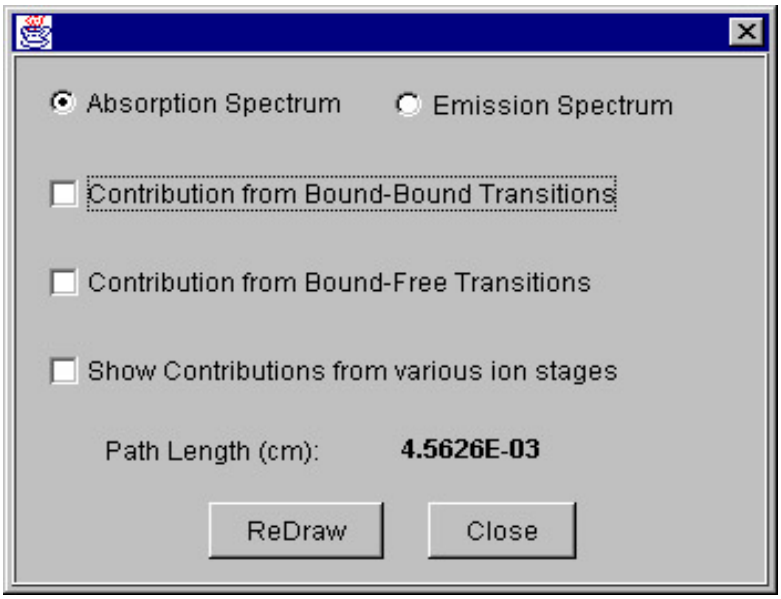


Fig. A.5.6 Spectrum control panel for viewing different components of the spectrum.

5. Implementation

The implementation of this module needs knowledge about the client/server techniques. First, we specify the interfaces that may be used in the program (see Section 4.3 in the thesis). The interfaces are implemented using CORBA. Then, the CORBA servers are deployed into the database. The CORBA clients are used as stubs in the application. For graphic user interfaces, we use inner frame to contain more than

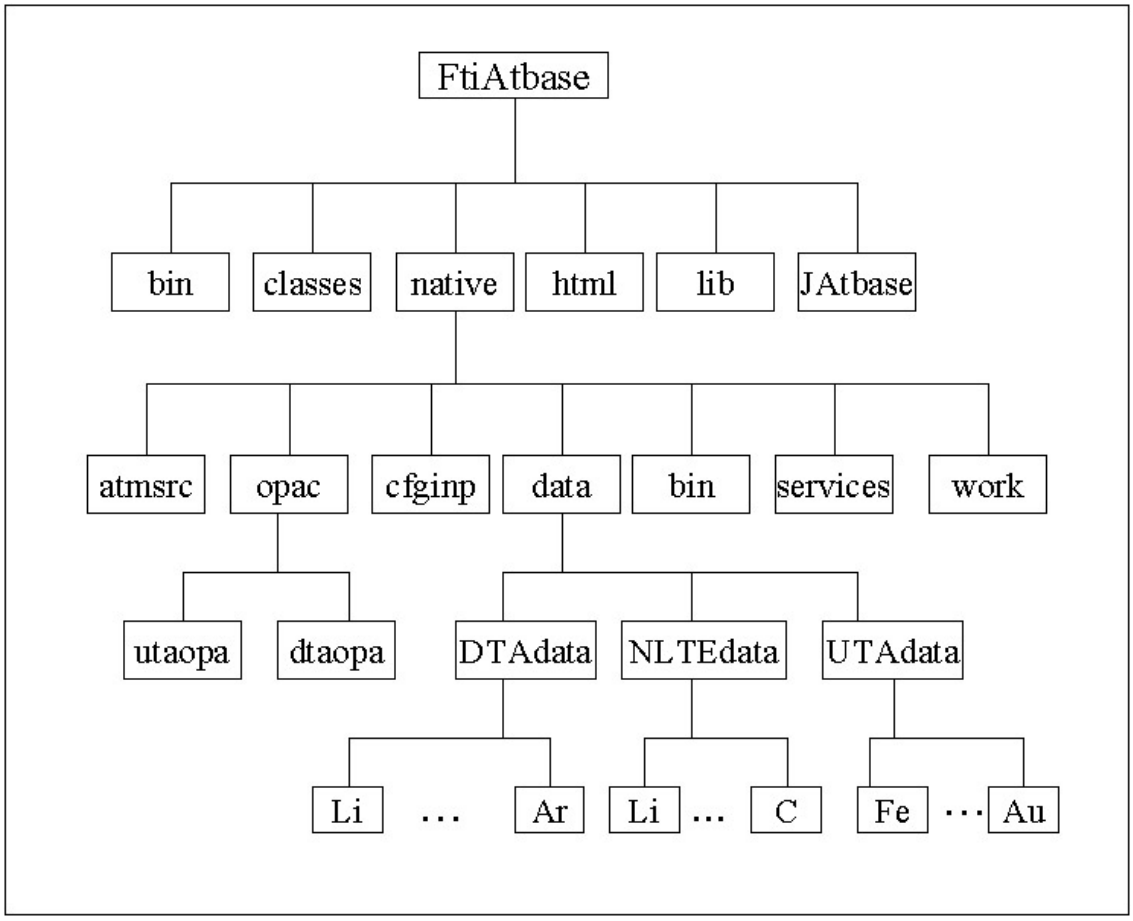
one graphs in the window. Detailed information about the GUI design is as following: *SpectraFrame* installs three panels: *SpectPlasma*, *ManipulatePanel*, and D*rawSpectraPanel.* These three panels are accessible from the static copy of itself. In its constructor, the *DrawSpectraPanel* is also registered by the *DrawManager*, which is in charge of adding the *DrawInnerFrame* object if no identical copy exists, or removing the *DrawInnerFrame* object if it is closed. The *DrawSpectraPanel* is the main plotting area that contains the inner frames. *DrawManager* has a static reference to the only one copy of *DrawSpectraFrame* object, which does the actual job to add the internal frame onto the *DrawSpectraFrame*. *DrawManager* also keeps track the plots that are added on the *DrawSpectraFrame*. *DrawInnerFrame* is the actual drawing object that does the job of contacting the CORBA client objects and drawing the graph on its own panel. Each *DrawInnerFrame* has a *PlasmaCondition* object which contains the plasma condition and the spectrum range that is constructed after the *Show* button is clicked.

5. Troubleshooting

None.

Appendix B

File structure of JAtbase native code



Appendix C

Summary of codes for atomic data calculations in FTI

Code	Model	Capability
ATBASE (serial)	DTA	▪ Detailed LSJ, LS coupling radiative atomic data (E, gf) ▪ Configuration interaction method with Hartree-Fock wavefunctions ▪ Bound-bound and bound free transitions ▪ Z = 1-18 (can be extended to medium Z)
	UTA	▪ Single-electron transition approximation ▪ Non-relativistic unresolved transition array method ▪ Bound-bound and bound-free transitions ▪ LS coupling ▪ Z = 1-79
EOSOPA (serial)	LTE	▪ Saha-Boltzmann distribution ▪ Voigt line shapes ▪ Pressure ionization effect ▪ Z = 1-79
	Non-LTE	▪ Collisional radiative equilibrium (CRE) equation ▪ Including electron collision and radiative atomic processes ▪ Z = 1-18
RSSOPA (parallel)	LTE UTA	▪ Single-electron transition approximation ▪ Relativistic unresolved transition array method ▪ Bound-bound and bound-free transitions ▪ JJ coupling ▪ Z = 1-79