Numerical Solution of Plasma Fluid Equations Using Locally Refined Grids

P. Colella
M.R. Dorr
D.D. Wake

January 26, 1997

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the Laboratory.
Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.
DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately own rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (615) 576-8401, FTS 626-8401

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161
Numerical Solution of Plasma Fluid Equations
Using Locally Refined Grids

Phillip Colella, Milo R. Dorr and Daniel D. Wake

January 26, 1997

Abstract

This paper describes a numerical method for the solution of plasma fluid equations on block-structured, locally refined grids. The plasma under consideration is typical of those used for the processing of semiconductors. The governing equations consist of a drift-diffusion model of the electrons and an isothermal model of the ions coupled by Poisson's equation. A discretization of the equations is given for a uniform spatial grid, and a time-split integration scheme is developed. The algorithm is then extended to accommodate locally refined grids. This extension involves the advancement of the discrete system on a hierarchy of levels, each of which represents a degree of refinement, together with synchronization steps to ensure consistency across levels. A brief discussion of a software implementation is followed by a presentation of numerical results.

1 Introduction

Many of the process steps performed in the manufacture of very large scale integrated (VLSI) circuits involve plasmas. Inductively coupled plasma (ICP) reactors represent one type of processing tool that utilizes high-density, low-pressure plasmas to satisfy the demanding process criteria resulting from the desire to create increasingly smaller device features on large wafers. Computational simulators of ICP reactors, such as the INDUCT package [13], can help equipment manufactures and process engineers understand the complex relationships among reactor and plasma parameters (e.g., reactor geometry, RF power, gas pressure, gas composition, electrode bias, densities, velocities, etc.) and process performance (e.g., etch rate, anisotropy, uniformity, selectivity, damage, etc.).

One of the primary difficulties in the development of computational models of plasma processes is the need to address problems associated with a wide range of temporal and spatial length scales. The high-density, low-pressure plasmas employed in ICP reactors are particularly challenging in this regard. Here, the difference between the dielectric relaxation time of the plasma and the reactor transit time for an ion can be many orders of magnitude. Moreover, due to the small Debye length of such plasmas, the sheath region over which much of the potential drop occurs is three or four orders of magnitude smaller than the reactor dimensions. Numerical methods must therefore be designed to accommodate large scale variations.

In this paper, we propose an algorithm for the solution of a system of plasma fluid equations on locally refined grids. We focus our attention on the discretization and solution of the fluid system on a spatial grid structure with multiple levels of resolution that are prescribed a priori. This is the first step toward a fully adaptive algorithm that will be described in a later report. Our approach represents an extension and modification of methodologies developed for neutral flows using Adaptive Mesh Refinement (AMR) to include the significant additional effects of electrostatic forces.
In Section 2, we present the system of equations to be solved. The plasma is considered to be a fluid consisting of two charged components, ions and electrons, coupled by Poisson’s equation. The motion of the electrons is described by a drift-diffusion model that includes electron temperature, and the ions are treated as an isothermal fluid. The system of equations is discretized in Section 3 using a combination of high order upwinding of the advective terms and an implicit treatment of the diffusive and electron drift flux terms. In Section 4, we extend the single grid algorithm to the case of block-structured locally refined grids. Beginning with a rectangular uniform grid, subregions are successively refined to create a hierarchy of refinement levels, each of which is a union of non-overlapping rectangles. The time evolution of the fluid systems is accomplished through the coordinated advancement of individual refinement levels, using the single grid algorithm of Section 3, together with several synchronization steps to ensure consistency among levels. An implementation of the algorithm is briefly described in Section 5, and numerical results are presented in Section 6.

2 The Plasma Model

The basic governing equations for a plasma fluid model are Maxwell’s equations along with moments of the Boltzmann equation. The plasma components are assumed to have a drifting, near-Maxwellian velocity distribution. The infinite set of moment equations is truncated and includes only the first three. These moments correspond to the conservation of particles, momentum and energy. This is the starting point for our model. A detailed development can be found in [9].

We wish to apply these equations to a two-species plasma of electrons and singly ionized ions. We first make some additional approximations. For the electrons, we make the drift-diffusion approximation for the electron directed velocity. This approximation is sometimes referred to as the zero mass approximation, but actually it is just the elimination of inertial terms which are negligible provided that electron plasma oscillations are not resolved. The result is the replacement of the electron momentum equation with an equation of state for the electron flux determined from the balance of forces

\[ n_e u_e = -\frac{e n_e}{m_e \nu_{en}} E - \frac{1}{m_e \nu_{en}} \nabla (n_e k T_e) \]  

(2.1)

where the first term on the right hand side is the drift flux and the second term is the diffusive flux. Using the electron continuity equation and (2.1), we also replace the electron energy equation by an equation for the internal energy \( \frac{1}{2} n_e k T_e \).

We assume that the ions are isothermal and have the same temperature as the neutral background gas, which is assumed to have constant density. We therefore do not need to retain the ion energy equation. These assumptions are made to simplify the model, but are not essential to the algorithm and implementation described here.

The coupling of external power to the plasma is via a deposition source term in the electron energy equation. We assume that this source term is known a priori, but in a full electromagnetic model it would be obtained through a self-consistent treatment of the source and plasma. For ICP applications, the high thermal conductivity of the plasma along with the small skin depth make the decoupling of the source and plasma a reasonable approximation provided the region of interest is more than a few skin depths away from the coils (typically a few centimeters). With this simplification and the absence of external fields, we may neglect magnetic fields. The electromagnetic behavior is then simply described by Poisson’s equation.

With these modifications, we are left with 6 scalar equations in two dimensions. They are the electron continuity and internal energy equations, the ion continuity and momenta equations and Poisson’s equation.
Table 2.1 defines the variables used in these equations.

### Electron Equations

\[
\frac{\partial n_e}{\partial t} + \nabla \cdot (u_e n_e) = \nu_{iz} n_e \tag{2.2}
\]

\[
\frac{\partial (n_e^3 kT_e)}{\partial t} + \nabla \cdot \left( n_e^5 \frac{kT_e u_e}{2} \right) = -en_e u_e \cdot E - \nabla \cdot \left( \frac{5 n_e kT_e}{2 m_e v_{eN}} \nabla kT_e \right) + P_{\text{ind}} - \left( \nu_{iz} \epsilon_{iz} + \nu_{elas} \epsilon_{elas} + \frac{1}{2} m_e u_e \cdot u_e \nu_{iz} \right) n_e. \tag{2.3}
\]

### Ion Equations

\[
\frac{\partial n_i}{\partial t} + \nabla \cdot (u_i n_i) = \nu_{iz} n_e \tag{2.4}
\]

\[
\frac{\partial (n_i u_i)}{\partial t} + \nabla \cdot (n_i u_i u_i) + \frac{kT_i}{m_i} \nabla n_i = \frac{q_m}{m_i} E = u_i (\nu_{iz} n_e - \nu_{iN} n_i) \tag{2.5}
\]

### Poisson's Equation

\[
\frac{\epsilon_0}{e} \nabla^2 \phi = n_e - n_i \tag{2.6}
\]
3 Single Grid Algorithm

3.1 Motivation

Before describing the single grid algorithm, we consider the plasma characteristics and mathematical difficulties which affect the choice of an algorithm.

The relatively slow motion of the ions allows some flexibility in the manner in which they are advanced. We do, however, want to retain high spatial accuracy since the ion flux is an important quantity for processing applications. If we explicitly treat the low order terms in (2.4) and (2.5), the ion equations form a nonlinear, vector conservation law. Such equations are well understood [5] and stable, second-order advection schemes may be applied.

The electron density and potential update is the most challenging part of the algorithm. Solving Poisson’s equation with explicitly advected electrons would force the use of an extremely small timestep due to the dielectric relaxation limit [2], [12]. A tight coupling of the electron density and potential is assured by including the drift flux with the Laplacian operator, forming a modified Poisson equation. This implicit treatment eliminates problems associated with dielectric relaxation timestep constraints.

The drift flux within the modified Laplacian operator depends on the electron density as well as the potential. The diffusive flux term depends on the electron density. To avoid simultaneously solving for the electron density and the potential, we approximate the density used for these terms. For the drift flux, the electron density is approximated at \( t^{n+1/2} \) on cell edges using a slope-limited, upwind Taylor-series expansion relative to the lagged drift velocity. This density approximation is denoted by \( n_e^{n+1/2} \). For the diffusive flux, we implicitly solve for an approximation at the new time and use it within a Crank-Nicholson discretization. This approximation is denoted by \( n_e^{n+1} \). The temperature variation is slow and is therefore lagged in these terms. The drift flux term is then approximated by

\[
n_e u_e \approx -\frac{c n_e^{n+1/2}}{m_e v_{en}^n} E^{n+1} - \frac{1}{2m_e v_{en}^n} \nabla (\hat{n}_e^{n+1} k T_e^n + n_e^n k T_e^n)
\]  

(3.7)

The internal energy equation involves a stiff heat flux term. This term should therefore be treated implicitly.

3.2 Time Splitting Scheme

The time splitting scheme contains 5 steps. Before detailing them, we briefly outline the single grid algorithm:

1. Explicitly calculate the flux divergence needed to update the ion data using the continuity equation. Defer the final calculation of the ion data until the electrons and potential have been advanced.

2. Predict an edge and time centered electron density as well as an implicitly calculated cell centered electron density at the new time. These will be used for the electron flux approximation (3.7).

3. Solve the modified Poisson equation for the new potential. Use the new electric field to update the electron density and flux.

4. Update the electron internal energy using the new electron data and electric field.
5. Perform a final update on the ion data using a temporally second-order accurate source term.

Our spatial discretization begins with the cell averaging of each of the continuous equations on the underlying uniform grid. This results in cell averaged dependent variables, and the divergence operator becomes a surface integral operator

\[ \nabla \cdot F \equiv \frac{1}{\text{cell volume}} \int F \cdot \hat{n} \, dS \]

With this interpretation, the usual differential notation is retained for convenience.

We can now describe the single grid algorithm. Given known values of all quantities at time \( t_n \), we wish to integrate (2.2)–(2.6) up to a new time \( t_{n+1} = t_n + \Delta t \). The timestep \( \Delta t \) is chosen to satisfy the CFL conditions imposed by the explicit steps in the algorithm. We proceed as follows:

**Step 1**

The update formula for the ion continuity equation is

\[
\frac{n_i^{n+1} - n_i^n}{\Delta t} = -\nabla \cdot (n_i u_i)^{n+1/2} + \frac{\nu_i^2}{2} \left( n_i^{n+1} + n_i^n \right). \tag{3.8}
\]

In the explicit method used to compute the flux divergence \((n_i u_i)^{n+1/2}\) the lagged field \( E^n \) and ionization source \( \nu_i^2 n_i^n \) are used. Since the time-centered ionization source represented by the last term in the right-hand side of (3.8) cannot be computed until \( n_i^{n+1} \) is known, the final computation of \( n_i^{n+1} \) is postponed until \( n_i^{n+1} \) has been computed. This allows the source term contribution to be temporally second order accurate. The reason we may defer the final update is that the ion density flux term \( \nabla \cdot (n_i u_i)^{n+1/2} \), is the only part of the ion advance required by the rest of the algorithm.

To perform the integration, we notice that the system of ion equations (2.4)–(2.5) can be written as a single vector equation

\[
\frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) + \frac{\partial}{\partial y} G(U) = H(U, E, n_e) \tag{3.9}
\]

where we have the following definitions:

\[
U = \begin{bmatrix} n_i \\ \Gamma_x \\ \Gamma_y \end{bmatrix}, \quad \Gamma_x = n_i u_x, \quad \Gamma_y = n_i u_y, \quad c = \sqrt{\frac{kT_i}{m_i}} \tag{3.10}
\]

\[
F(U) = \begin{bmatrix} \Gamma_x \\ \Gamma_x \frac{\Gamma_x^2}{n_i} + c^2 n_i \\ \frac{\Gamma_x \Gamma_y}{n_i} \end{bmatrix}, \quad G(U) = \begin{bmatrix} \Gamma_x \\ \Gamma_x \frac{\Gamma_y}{n_i} \\ \frac{\Gamma_y}{n_i} + c^2 n_i \end{bmatrix} \tag{3.11}
\]

\[
H(U, E, n_e) = \begin{bmatrix} \nu_{ix} n_e \\ \nu_{ix} \Gamma_x - \nu_{ix} n_e \Gamma_x \Gamma_x/n_i \\ \nu_{ix} \Gamma_y - \nu_{ix} n_e \Gamma_y \Gamma_y/n_i \end{bmatrix} \tag{3.12}
\]

where \( n_i \) is the ion density, \( u_x \) and \( u_y \) are the ion velocities, \( T_i \) is the ion temperature and \( m_i \) is the ion mass. This is just the isothermal Euler equation with the inclusion of a source term. Such systems have been studied and methods developed to solve them even for stiff source terms [11].

We use a second order accurate, unsplit Godunov method as described in [5] to solve this system and obtain \( \nabla \cdot (n_i u_i)^{n+1/2} \).
Step 2

To begin the electron advance, substitute (2.1) into (2.2) to obtain

\[
\frac{\partial n_e}{\partial t} = -\nabla \cdot (n_e \vec{u}_e) + \nabla \cdot \left( \frac{1}{m_e \nu_{en}} \nabla (n_e kT_e) \right) + \nu_{iz} n_e
\]  

(3.13)

where \( \vec{u}_e \equiv -eE/m_e \nu_{en} \). First, the edge and time centered electron prediction \( n_e^{n+1/2} \) is calculated. We then substitute the flux approximation (3.7) to obtain an equation for \( \dot{n}_e^{n+1} \). This is discretized as follows:

\[
\frac{\dot{n}_e^{n+1} - n_e^n}{\Delta t} = -\nabla \cdot (n_e^{n+1/2} \vec{u}_e^n) 
+ \frac{1}{2} \left[ \nabla \cdot \left( \frac{1}{m_e \nu_{en}^n} \nabla \dot{n}_e^{n+1} kT_e^n \right) + \nabla \cdot \left( \frac{1}{m_e \nu_{en}^n} \nabla n_e^{n+1} kT_e^n \right) \right] + \frac{\nu_{iz}^n}{2} (\dot{n}_e^{n+1} + n_e^n) 
\]

(3.14)

\[
\dot{\vec{u}}_e^n \equiv -\frac{e}{m_e \nu_{en}^n} \vec{E}^n. 
\]

(3.15)

Collecting terms involving \( \dot{n}_e^{n+1} \) on the left-hand side yields a linear system that can be solved for this quantity.

Step 3

Now that we have predictions for the electron density, we can form the modified Poisson equation. Difference the constraint (2.6) to obtain

\[
\frac{\varepsilon_0}{e} \nabla^2 \phi^{n+1} - \nabla^2 \phi^n = \frac{n_e^{n+1} - n_e^n}{\Delta t} - \frac{n_i^{n+1} - n_i^n}{\Delta t}. 
\]

(3.16)

Use the continuity equation to replace the first term on the right-hand side by

\[
\frac{n_e^{n+1} - n_e^n}{\Delta t} = -\nabla \cdot \left( n_e^{n+1/2} \dot{n}_e^{n+1} \right) 
+ \frac{1}{2} \left[ \nabla \cdot \left( \frac{1}{m_e \nu_{en}^n} \nabla \dot{n}_e^{n+1} kT_e^n \right) + \nabla \cdot \left( \frac{1}{m_e \nu_{en}^n} \nabla n_e^{n+1} kT_e^n \right) \right] + \frac{\nu_{iz}^n}{2} (n_e^{n+1} + n_e^n) 
\]

(3.17)

where

\[
\dot{\vec{u}}_e^{n+1} \equiv -\frac{e}{m_e \nu_{en}^n} \vec{E}^{n+1} = \frac{e}{m_e \nu_{en}^n} \nabla \phi^{n+1}. 
\]

(3.18)

Replace the second term on the right-hand side of (3.16) by (3.18). Collecting all terms involving \( \phi^{n+1} \) on the left-hand side and multiplying by \( \Delta t \), we obtain the modified Poisson equation

\[
\nabla \cdot \left[ \left( \frac{\varepsilon_0}{e} + \frac{en_e^{n+1/2} \Delta t}{m_e \nu_{en}^n} \right) \nabla \phi^{n+1} \right] = 
\]

(3.19)

\[
\frac{\varepsilon_0}{e} \nabla^2 \phi^n + \Delta t \nabla \cdot (n_i \dot{u}_i)^{n+1/2} + \frac{1}{2} \left[ \nabla \cdot \left( \frac{\Delta t}{m_e \nu_{en}^n} \nabla \dot{n}_e^{n+1} kT_e^n \right) + \nabla \cdot \left( \frac{\Delta t}{m_e \nu_{en}^n} \nabla n_e^{n+1} kT_e^n \right) \right] 
\]

which is an elliptic problem that can be solved for \( \phi^{n+1} \). Using (3.17), the value of \( n_e^{n+1} \) is then obtained via the formula

\[
n_e^{n+1} = \left( 1 - \frac{\nu_{iz}^n \Delta t}{2} \right)^{-1} \left\{ -\nabla \cdot (n_e^{n+1/2} \Delta t \dot{n}_e^{n+1}) 
+ \frac{1}{2} \left[ \nabla \cdot \left( \frac{\Delta t}{m_e \nu_{en}^n} \nabla \dot{n}_e^{n+1} kT_e^n \right) + \nabla \cdot \left( \frac{\Delta t}{m_e \nu_{en}^n} \nabla n_e^{n+1} kT_e^n \right) \right] + \left( 1 + \frac{\nu_{iz}^n \Delta t}{2} \right) n_e^n \right\} 
\]

(3.20)
Step 4

We discretize (2.4) as

\[
\frac{3 n_e^{n+1} k T_e^{n+1} - n_e^n k T_e^n}{\Delta t} + \nabla \cdot \left( \frac{5}{2} n_e^{n+1/2} k T_e^{n+1/2} \right) = -e \Gamma_e^{n+1/2} \cdot E^{n+1/2}
\]

\[+ \frac{5}{2} \nabla \cdot \left( \frac{n_e^{n+1/2} k T_e^{n+1/2}}{m_e e n_e} \nabla (k T_e^{n+1}) \right)
\]

\[+ P_{ind} - \left( \nu_{iz}^n + \nu_{elas}^{n} + \frac{1}{2} m_e u_e^n \cdot u_e^n \right) n_e^{n+1/2}
\]

(3.21)

\[
T_e^{n+1/2} \text{ is a slope-limited, Taylor-series prediction of the edge temperature at } t_{n+1/2} \text{ upwinded relative to the velocity } u_e^n \text{ where}
\]

\[
E^{n+1/2} = -\nabla (\phi^{n+1} + \phi^n)/2,
\]

(3.22)

\[
\bar{u}_e^{n+1/2} = -\frac{e}{m_e e n_e} E^{n+1/2}
\]

(3.23)

\[
u_e^{n+1/2} = \bar{u}_e^{n+1/2} - \frac{1}{n_e^{n+1/2} m_e e n_e} \nabla n_e^{n+1/2} k T_e^{n+1/2}
\]

(3.24)

\[
\Gamma_e^{n+1/2} = n_e^{n+1/2} u_e^{n+1/2}.
\]

(3.25)

Collecting all terms involving \(T_e^{n+1}\) on the left-hand side, we obtain a linear system that can be solved for this quantity.

Step 5

At this point, all that remains is the final update of the ion data. To do this, we must evaluate the source term \(H(U, E, n_e)\) in equation (3.9). At the start of the timestep we have ion data

\[
U^n = \begin{bmatrix} n_i^n \\ \Gamma_i^n \\ \Gamma_e^n \\ \Gamma_y^n \end{bmatrix}, \quad n_e = n_e^n, \quad E = E^n
\]

(3.26)

At the end of Step 3, we have provisional values for the ion data and final values for the electron density and electric field

\[
\tilde{U}^{n+1} = \begin{bmatrix} \hat{n}_i^{n+1} \\ \hat{\Gamma}_i^{n+1} \\ \hat{\Gamma}_e^{n+1} \\ \hat{\Gamma}_y^{n+1} \end{bmatrix}, \quad \bar{n}_e = n_e^{n+1}, \quad E = E^{n+1}
\]

(3.27)

We then average the source term evaluated with these two states. This is much like Heun’s method [7] and is second order accurate in time. With this source term we can apply the final update to the conservation law (3.9)

\[
U^{n+1} = U^n - \frac{\Delta t}{\Delta x} [F(\tilde{U}_R^{n+1/2}) - F(\tilde{U}_L^{n+1/2})] - \frac{\Delta t}{\Delta y} [G(\tilde{U}_T^{n+1/2}) - G(\tilde{U}_B^{n+1/2})] +
\]

\[+ \frac{\Delta t}{2} [H(U^n, E^n, n_e^n) + H(\tilde{U}^{n+1}, E^{n+1}, n_e^{n+1})]
\]

(3.28)

where \(\tilde{U}_R^{n+1/2}, \tilde{U}_L^{n+1/2}, \tilde{U}_T^{n+1/2}\) and \(\tilde{U}_B^{n+1/2}\) are Riemann solutions for the evaluation of time centered fluxes on the right, left, top and bottom cell edges, respectively.
3.3 Boundary Conditions

For the potential $\phi$, Dirichlet boundary conditions are imposed corresponding to an applied voltage at the reactor walls.

For the electron number density $n_e$ and temperature $T_e$, the boundary conditions are derived from an estimate of the electron flux $n_e u_e$. Consider a spatial cell with an edge on the problem boundary. Let $\Gamma_b$ denote the outward normal component of $n_e u_e$ at the center of the boundary edge, and let $\phi_b$ denote the applied voltage there. Let $\phi$, $n_e$ and $T_e$ denote the potential, electron number density and electron temperature at the cell center, respectively. Of all the electrons at the cell center, only those moving toward the boundary with energies greater than $e|\phi_b - \phi|$ can overcome the potential barrier. Using the Maxwellian distribution

$$ f(w) = \left( \frac{m_e}{2\pi kT_e} \right)^{\frac{3}{2}} \exp \left( -\frac{m_e w^2}{2kT_e} \right) $$

we therefore propose that

$$ \Gamma_b = n_e \bar{\omega}_e(kT_e, \phi) $$

where

$$ \bar{\omega}_e(kT_e, \phi) = \int_{\frac{2e(\phi_b - \phi)}{m_e}}^{\infty} \frac{1}{\sqrt{2\pi m_e kT_e}} \exp \left( -\frac{e|\phi_b - \phi|}{kT_e} \right) dw $$

Let $n_e^{n+1/2}$ again denote the Taylor-series-predicted density at the boundary. We can then require that at the problem boundary

$$ n_e^{n+1/2} \bar{\omega}_e^k = \frac{1}{m_e \nu_{en}} \nabla (n_e^k kT_e^n) = n_e^k \bar{\omega}_e^k(kT_e^n, \phi^k) \quad k = n, n + 1 $$

which provides boundary conditions for the corresponding terms in (3.14), (3.20) and (3.21).

The solution of the linear system represented by (3.21) requires a boundary condition for the heat flux

$$ q_e \equiv \frac{5}{2} \frac{n_e kT_e}{m_e \nu_{en}} kT_e $$

We assume the following boundary condition on the energy flux

$$ \frac{5}{2} n_e kT_e u_e + q_e = 2n_e u_e kT_e $$

from which the boundary condition for $q_e$ is obtained.

For the ions, we use Godunov's method which requires the solution of Riemann problems at all cell edges during a timestep. These Riemann solutions give the nonlinear equivalent of upwind solutions which are used to evaluate the flux functions $F$ and $G$ in (3.9). On interior edges, predictions are obtained on both sides of each edge for these Riemann problems. Instead of solving such a problem at physical boundaries, we instead assume that the prediction coming from the interior of the domain is the solution to the Riemann problem.

This has an interesting physical interpretation. Provided the ions are flowing outward at a velocity greater than the sound speed $\sqrt{kT_i/m_i}$, then the choice of the interior prediction as the Riemann solution is correct. Since the ion velocity at the wall is almost always supersonic, this should be an accurate boundary condition.
Figure 4.1: Sample three-level locally refined grid structure.

4 Solution of the Fluid Equations on Locally Refined Grids

The preceding description of the spatial and temporal discretization assumed a uniform spatial grid with timesteps selected to satisfy the stability requirements of the explicit integration of the ion equations and electron drift. In this section, we generalize the approach to enable the use of locally refined grids.

Our spatial grid structure consists of a hierarchy of refinement levels indexed by $\ell = 0, \ldots, \ell_{\text{max}}$. Beginning with the physical domain, assumed to be a rectangle, a uniform grid $G_0$ is selected as the first and coarsest grid in the hierarchy. For $\ell = 1, \ldots, \ell_{\text{max}}$, refinement level $\ell$ is a union $G_\ell$ of non-overlapping uniform grids, each of which is obtained by refining a rectangular subset of $G_{\ell-1}$ using a refinement ratio $n_{\text{ref},\ell-1}$. The refinement ratios $n_{\text{ref},\ell}$ are assumed to be powers of two, and each is a constant parameter on their respective levels; i.e., the mesh sizes of all grids at a particular refinement level are identical. The location of grids is arbitrary except for the requirement that each $G_\ell$ be properly contained in the region covered by $G_{\ell-1}$ (except at the physical boundary where all levels can be refined up to the edge). Figure 4.1 depicts a sample grid hierarchy.

The time integration described in Section 3 proceeds in several steps, each of which falls generally into one of the following categories:

- **Explicit integration of a system of hyperbolic conservation laws.** The integration of the ions in Steps 1 and 5, as well as the prediction of the edge-centered electron densities and energy fluxes in Steps 2 and 4 belong to this category.

- **Implicit integration of stiff parabolic equations.** This characterizes the computation of electron density prediction $n_e^{n+1}$ in Step 2, the potential in Step 3 and the internal electron energy in Step 4.
The generalization of the single-grid algorithm of Section 3 to the locally refined grid structure introduced above therefore requires the ability to solve these two types of subproblems on such a hierarchy. To maintain accuracy and conservation, a critical issue to be addressed in each case is the enforcement of matching conditions at the interfaces between coarse and fine grids. The fact that matching conditions for purely hyperbolic or purely parabolic problems are well understood, and that extensible software libraries (described in the following section) have been developed to help implement the solution of these problems on the locally refined grids considered here, was one of the primary motivations for adopting the particular time splitting employed in the algorithm of Section 3.

To perform the explicit integration of the hyperbolic systems, we adopt the basic approach of [3]. The key elements in the present context are the techniques presented for interpolating boundary conditions (solutions obtained on coarser levels are temporally and spatially interpolated to provided boundary values for the integration of finer levels) as well as the preservation of conservation by careful flux accounting at coarse-fine interfaces and the averaging and redistribution of fine-grid-computed quantities to coarser grids. An important observation is that no significant changes are required to the single-grid integration algorithm to accommodate these additional requirements.

The parabolic systems are solved by using backward Euler differencing in time, as indicated by the time centerings in the equation (3.14) determining \( n_{e}^{n+1} \), the equation (3.20) for \( \phi^{n+1} \), and the equation (3.21) for \( n_{e}^{n+1} kT_{e}^{n+1} \). In the first and third cases, this yields a linear system to be solved at each timestep. The potential equation to be solved is nonlinear due to the boundary condition (3.32), but its solution via Newton iteration ultimately yields a Jacobian operator to be inverted on the grid hierarchy. In all three cases, the linear operators are second-order, self-adjoint and elliptic. This imposes certain compatibility conditions at the interfaces between coarse and fine grids. For the equation (3.20), we require continuity of the potential at such coarse-fine interfaces, as well as continuity of the normal component of the field. For the equation (3.21), we require continuity of the temperature and the normal component of the heat flux. Moreover, we must ensure that the coarse and fine grids are coupled in such a way that fine grid accuracy is not unnecessarily reduced by the coupling with the coarse grid. To address these issues, we generalize the approach described in [10] for Poisson's equation to the variable coefficient cases represented by (3.20) and (3.21). This is a multilevel algorithm that iterates over the grid hierarchy in a multigrid-like fashion, taking care to enforce the required elliptic matching conditions and using high-order interpolation to preserve accuracy at coarse-fine boundaries. The linear systems and the algorithm of [10] are described in more detail in Sections 4.1 through 4.3 below.

For the remainder of this section, it will be convenient to have the following terminology and notation. For \( \ell = 0, \ldots, \ell_{\text{max}} - 1 \), let \( G_{t}^{\ell} \) denote the portion of \( G_{t} \) that has been refined to obtain \( G_{t+1} \). The subregion \( G_{t}^{\ell+1} \) can be regarded as a projection of \( G_{t+1} \) into \( G_{t} \), or simply as the portion of \( G_{t} \) that is "covered" by \( G_{t+1} \). For \( \ell = 0, \ldots, \ell_{\text{max}} - 1 \), let \( G_{t}^{u} \equiv G_{t} \setminus G_{t}^{\ell} \) denote the "uncovered" portion of \( G_{t} \). We also let \( G_{t}^{u_{\text{max}}} \equiv G_{t_{\text{max}}} \). It will be convenient to reference specifically the edges of cells in the grids \( G_{t} \). We therefore let \( H_{t}^{\ell} \) denote the edges of cells in \( G_{t}^{\ell} \), and let \( H_{t}^{u} \) denote the grid of edges of cells in \( G_{t}^{u} \), excluding those belonging to \( H_{t}^{\ell} \). Note that with these definitions, edges in \( H_{t}^{u} \) are covered by edges in \( H_{t+1}^{u} \cup H_{t+1}^{u_{\text{max}}}, \) the edges of cells in \( G_{t+1} \).

We seek solutions on the union of the uncovered grids

\[
G_{\text{composite}} \equiv \bigcup_{\ell=0}^{\ell_{\text{max}}} G_{t}^{u}
\]

which we call the composite grid. To enable these solutions, and other multilevel quantities that are individually evolved level by level, to be composed from the "uncovered" parts of single-level quantities, care must be taken to ensure that the single-level quantities represent appropriate
coarsenings of the composite values on finer levels. This is accomplished using various spatial averages, denoted $\langle \cdot \rangle_S$, such as cell-volume weighting to conservatively average densities. Timestep-weighted temporal averaging, denoted $\langle \cdot \rangle_T$, is also required. Averaging with respect to both time and space will be denoted by $\langle \cdot \rangle$. We must also ensure that edge-centered quantities such as fluxes are conservatively defined at the coarse–fine interfaces. Both notationally as well as in the implementation described in the following section, we employ the concept of flux registers introduced in [3] to contain the differences in the fine and coarse grid computed fluxes (and other edge-centered quantities such as electric fields) at the coarse–fine interfaces. Flux registers are primarily used to update the value of a cell-centered quantity on cells of $G_0^n$ that have one or more edges contained in the coarse–fine interface with $G_{t+1}^n$. In that case, the flux register would contain the difference between the more accurate fine grid fluxes computed on $G_{t+1}^n$ and the fluxes on $G_t$ originally used to compute the cell-centered quantity. The update is accomplished by “refluxing”, i.e. adding the result of applying a reflux divergence operator $D_R$ to the flux register $\delta F$, formally defined as

$$D_R(\delta F) = \frac{1}{\text{cell volume}} \oint \delta F \cdot \hat{n} \, dS \text{ on cells in } G_t^{n}$$

where we assume that $\delta F = 0$ on cell boundaries not contained in the coarse–fine interface. We remark that $D_R(\delta F)$ is not required to update cell-centered data on the covered region $G_t^n$ (and is therefore not defined there), since this is accomplished by averaging data from the finer level $G_{t+1}$ in a separate step.

We now describe the multilevel algorithm. For simplicity, we initially assume a fixed two-level grid structure where level 1 is obtained by refining some portion of level 0 using a refinement ratio $n_{ref}$.

1. Determine timesteps $\Delta t_0^{\text{max}}$ and $\Delta t_1^{\text{max}}$ for which the explicit parts of the single–level algorithm are stable on level 0 and level 1, respectively. Then select timesteps $\Delta t_0$ and $\Delta t_1$ such that $\Delta t_0 = \Delta t_1 n_{ref}$ and $\Delta t_\ell \leq \Delta t_\ell^{\text{max}}$ for $\ell = 0, 1$. Assume that the dependent variables $n_i, \Gamma_x, \Gamma_y, n_e, T_e$ and $\phi$ are known at some starting time, $t_{\text{init}}$.

2. Apply the single–level algorithm to advance the dependent variables on level 0 from time $t_{\text{init}}$ to $t_{\text{init}} + \Delta t_0$. The boundary values required for these quantities are obtained from physical boundary conditions. During the integration, we accumulate and store fluxes and fields at cell edges lying on the coarse–fine interface between level 0 and level 1. These quantities are used in correction steps to follow.

3. Apply the single–level algorithm to advance the dependent variables on level 1 from $t_{\text{init}}$ to $t_{\text{init}} + \Delta t_0$ by taking $n_{ref}$ timesteps of size $\Delta t_1$. The boundary values required for these quantities are obtained either from physical boundary conditions (on grid boundaries contained in the physical domain boundary) or from spatial and temporal interpolation from coarser levels (on grid boundaries in the problem interior). During this integration, we again accumulate fluxes and fields on the coarse–fine grid boundaries.

4. Correct the level 0 electron density and the level 0 ion density and fluxes to account for the fact that the level 1 integration steps will have obtained more accurate values (due to the increased temporal and spatial resolution) of these quantities. On $G_0^n$, this update is accomplished using a volume-weighted average of the corresponding fine grid quantities. On $G_0^n$, the electron densities and ion densities/fluxes are updated by refluxing the differences stored in flux registers during the advance step. At this time, we also update the level 0 electron temperature by applying the same averaging and refluxing operations to the internal electron temperature $3n_e kT_e/2$ from which $kT_e$ is then derived.
5. Next, we correct the potential \( \phi \) on the composite grid to account for the fact that the refluxing of electrons and ions into or out of \( G_0^n \) in the preceding steps represents a redistribution of charge. We must also ensure that the corrected composite potential and associated field are both continuous at the coarse-fine interface, and that any potential adjustment made to achieve this goal is accompanied by a redistribution of charge using the corresponding drift flux. A correction meeting all of these criteria is obtained by solving a composite problem with level operators similar to that of (3.20), but with a specially constructed source term. In a manner analogous to the updating of electron density (3.21) following the determination of the potential in (3.20) in the single-grid algorithm, we update the composite electron density following the composite potential correction using the corresponding electron drift flux. This step is described in more detail in Section 4.1.

6. Finally, we correct the composite electron temperature (i) to account for the refluxing of internal electron energy following the integration of level 1, (ii) to account for the change in internal electron energy due to the redistribution of electrons in the preceding potential and drift flux correction step, and (iii) to ensure continuity of both the electron temperature and heat flux at the coarse-fine interface. As with the potential, a single temperature correction achieving these goals can be obtained by solving a composite problem, which is discussed in more detail in Section 4.2.

For more than two grid levels, the algorithm can be implemented recursively, with the composite potential and temperature corrections occurring at the end of timesteps at each level (except the finest level) and involving all finer levels. Figure 4.2 schematically depicts the timestep advance and composite synchronization schedule for a 4-level problem with \( n_{ref,0} = n_{ref,1} = 2 \) and \( n_{ref,2} = 4 \). The horizontal arrows denote single-level time advance steps, while the vertical arrows denote synchronization of the composite grid.

In the next two subsections, we describe the synchronization of the potential and electron temperature following advancement on individual levels. Both of these synchronizations will require the solution of composite systems, which is the topic of Section 4.3.
4.1 Composite Potential Correction

After level $\ell = \ell_{\text{base}}$ is advanced one timestep using the single-level algorithm of Section 3, the following equation will hold on $G^u_\ell$

$$\frac{\varepsilon_0}{e\Delta t} (\nabla \cdot \tilde{E}^{n+1} - \nabla \cdot E^n) = \nabla \cdot \tilde{\Gamma}^{n+1/2}_e - \nabla \cdot \tilde{\Gamma}^{n+1/2}_i \tag{4.36}$$

where $\tilde{\Gamma}^{n+1/2}_e$ and $\tilde{\Gamma}^{n+1/2}_i$ are the total electron and ion fluxes, respectively,

$$\nabla \cdot E^n = -\nabla^2 \phi^n + D_R(\delta F^n_{\text{field}}) \tag{4.37}$$

$$\nabla \cdot \tilde{E}^{n+1} = -\nabla^2 \tilde{\phi}^{n+1} + D_R(\delta F^{n+1}_{\text{field}}) \tag{4.38}$$

and

$$\delta F^n_{\text{field}} = \left( E^{m+1}_t \right)_S - E^n_\ell \tag{4.39}$$

on the coarse-fine interface, where $m_n$ denotes the number of level $\ell + 1$ timesteps taken to reach time $t = t^n_\ell$, i.e., $t^{m+1}_\ell = t^n_\ell$. After levels $\ell = \ell_{\text{base}} + 1, \ldots, \ell_{\text{max}}$ are advanced to the same time as level $\ell_{\text{base}}$, the field $\tilde{E}^{n+1}$ can be replaced by a more accurate value $\tilde{E}^{n+1}$ computed using the new fine grid potentials such that on $G^u_\ell$

$$\nabla \cdot \tilde{E}^{n+1} = -\nabla^2 \tilde{\phi}^{n+1} + D_R(\delta \tilde{F}^{n+1}_{\text{field}}) \tag{4.40}$$

where

$$\delta \tilde{F}^{n+1}_{\text{field}} = -\left( \nabla \tilde{\phi}^{m+1}_t \right)_S + \nabla \tilde{\phi}^{n+1}_\ell \tag{4.41}$$

on the coarse-fine interface. We now seek a potential correction $\delta \phi$ such that

$$\frac{\varepsilon_0}{e\Delta t} (\nabla \cdot E^{n+1} - \nabla \cdot E^n) = \nabla \cdot \Gamma^{n+1/2}_e - \nabla \cdot \tilde{\Gamma}^{n+1/2}_i \tag{4.42}$$

on $G^u_\ell$, where

$$\nabla \cdot E^{n+1} = \nabla \cdot \tilde{E}^{n+1} - \nabla^2 \delta \phi^{n+1} + D_R(\delta F^{n+1}_{\text{field}}) \tag{4.43}$$

$$\nabla \cdot \tilde{\Gamma}^{n+1/2}_e = \nabla \cdot \tilde{\Gamma}^{n+1/2}_e + \nabla \cdot \alpha \nabla \delta \phi^{n+1} + D_R(\delta F^{n+1}_{\text{drift}}) \tag{4.44}$$

$$\nabla \cdot \tilde{\Gamma}^{n+1/2}_i = \nabla \cdot \tilde{\Gamma}^{n+1/2}_i + D_R(\delta F^{n+1}_{\text{drift}}) \tag{4.45}$$

and

$$\delta F^{n+1}_{\text{field}} = -\left( \nabla \delta \phi^{n+1}_t \right)_S + \nabla \delta \phi^{n+1}_\ell \tag{4.47}$$

$$\delta F^{n+1}_{\text{drift}} = \left( \Delta t \alpha_1 \alpha_{t+1} \nabla \delta \phi^{n+1}_t \right)_S - \Delta t \alpha_t \nabla \delta \phi^{n+1}, \tag{4.48}$$

on the coarse-fine interface, and with

$$\alpha_t = \begin{cases} \frac{m_n^{n+1/2}}{e_m \nu_{eN}} & \text{on } H^u_\ell \\
\langle \alpha_{t+1} \rangle & \text{on } H^u_\ell \tag{4.49} \end{cases}$$

Subtracting (4.36) from (4.42) gives the system be solved for corrections $\delta \phi_t$, $\ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}}$ on the composite grid

$$\nabla \cdot \left( \frac{\varepsilon_0}{e} + \Delta t \alpha_t \right) \nabla \delta \phi_t - \frac{\varepsilon_0}{e} D_R(\delta F^{n+1}_{\text{field}}) + D_R(\delta F^{n+1}_{\text{drift}}) = \frac{\varepsilon_0}{e} \left[ D_R(\delta \tilde{F}^{n+1}_{\text{field}}) - D_R(\delta F^n_{\text{field}}) \right] \tag{4.50}$$

$$+ D_R(\delta F^{n+1}_{\text{drift}}) - D_R(\delta F^{n+1}_{\text{drift}}) \text{ on } G^u_\ell \text{ for } \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} - 1$$
\[ \nabla \cdot \left( \frac{\rho_0}{e} + \Delta t_\ell \alpha_\ell \right) \nabla \delta \phi_\ell = 0, \quad \text{on } G_\ell \text{ for } \ell = \ell_{\text{max}}. \quad (4.51) \]

The final electron density update then becomes

\[ n^{n+1}_{e,\ell} = \begin{cases} n^{n+1}_{e,\ell} - \Delta t_\ell \nabla \cdot \alpha_\ell \nabla \delta \phi_\ell & \text{on } G^u_\ell \text{ for } \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} - 1 \\ n^{n+1}_{e,\ell_{\text{max}}} - \Delta t_{\ell_{\text{max}}} \nabla \cdot \alpha_{\ell_{\text{max}}} \nabla \delta \phi_{\ell_{\text{max}}} & \text{on } G_{\ell_{\text{max}}} \text{ for } \ell = \ell_{\text{max}} \\ n^{n+1}_{e,\ell+1} & \text{on } G^f_\ell \text{ for } \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} - 1. \end{cases} \quad (4.52) \]

The final electric field is obtained as

\[ E^{n+1}_\ell = \begin{cases} E^{n+1}_\ell - \nabla \delta \phi_\ell + \delta F^{n+1}_{\text{field}} & \text{on } H^u_\ell \text{ for } \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} - 1 \\ E^{n+1}_{\ell_{\text{max}}} - \nabla \delta \phi_{\ell_{\text{max}}} & \text{on } H^u_{\ell_{\text{max}}} \text{ for } \ell = \ell_{\text{max}} \\ \left< E^{n+1}_{\ell+1} \right>_{\text{ave}} & \text{on } H^f_\ell \text{ for } \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} - 1. \end{cases} \quad (4.53) \]

### 4.2 Composite Temperature Correction

After the advance steps, we have on each level \( \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} \)

\[ \frac{3 n^{n+1}_{e,\ell} k T^{n+1}_{e,\ell} - n^n_{e,\ell} k T^n_{e,\ell}}{\Delta t_\ell} + \nabla \cdot \left( \frac{5}{2} n^{n+1/2}_{e,\ell} k T^{n+1/2}_{e,\ell} \right) = 5 \frac{\nabla \cdot \left( n^{n+1/2}_{e,\ell} k T^{n+1/2}_{e,\ell} \right)}{m_e \nu_{\text{en}}} \nabla k \phi^{n+1}_{e,\ell} + P_{\text{net},\ell} - \frac{1}{\Delta t_\ell} D_R(\delta F_{\text{energy}}) + \frac{1}{\Delta t_\ell} D_R(\delta F_{\text{heat}}) \quad (4.54) \]

where

\[ P_{\text{net},\ell} = -e^{n+1/2}_\ell E^{n+1/2}_\ell \]

\[ + P_{\text{ind},\ell} \left( \nu^n_{iz} \nu_{iz} + \nu^n_{\text{elas}} \nu_{\text{elas}} + \frac{1}{2} m_e u^n_{e,\ell} \cdot u^n_{e,\ell} \right) n^{n+1/2}_{e,\ell} \quad (4.55) \]

and

\[ \delta F_{\text{energy}} = -\left( \sum_{k=m_{n}}^{m_{n+1}} \left( \frac{5}{2} \nabla \cdot \left( \Delta t_{\ell + \frac{1}{2}} (k T^{k+1/2}_{e,\ell+1}) \right) \right) + \frac{5}{2} \nabla \cdot \left( \Delta t_{\ell + \frac{1}{2}} (k T^{k+1/2}_{e,\ell+1}) \right) \right) \quad (4.56) \]

\[ \delta F_{\text{heat}} = -\left( \sum_{k=m_{n}}^{m_{n+1}} \frac{\Delta t_{\ell + \frac{1}{2}} (k T^{k+1/2}_{e,\ell+1})}{m_e \nu_{\text{en},\ell+1}} \nabla k T^{n+1}_{e,\ell+1} \right) + \frac{\Delta t_{\ell + \frac{1}{2}} (k T^{n+1/2}_{e,\ell+1})}{m_e \nu_{\text{en},\ell}} \nabla k T^{n+1}_{e,\ell} \quad (4.57) \]

unless \( \ell = \ell_{\text{max}} \), in which case \( \delta F_{\text{energy}} = \delta F_{\text{heat}} = 0 \).

After the composite solve for the potential correction described in Section 4.1, \( \hat{n}^{n+1}_{e,\ell} \) has been updated to \( n^{n+1}_{e,\ell} = \hat{n}^{n+1}_{e,\ell} + \delta n_{e,\ell} \) and \( \hat{k}^{n+1/2}_{\ell} \) has been updated to \( k^{n+1/2}_{\ell} = \hat{k}^{n+1/2}_{\ell} + \delta k_{\ell} \) for \( \ell = \ell_{\text{base}}, \ldots, \ell_{\text{max}} \). We also want to update \( k^{n+1}_{\ell} \) to \( k^{n+1}_{\ell} = k^{n+1}_{\ell} + \delta k_{\ell} \) satisfying

\[ \frac{3 n^{n+1}_{e,\ell} k T^{n+1}_{e,\ell} - n^n_{e,\ell} k T^n_{e,\ell}}{\Delta t_\ell} + \nabla \cdot \left( \frac{5}{2} k T^{n+1/2}_{e,\ell} \right) = 5 \frac{\nabla \cdot \left( n^{n+1/2}_{e,\ell} k T^{n+1/2}_{e,\ell} \right)}{m_e \nu_{\text{en}}} \nabla k T^{n+1}_{e,\ell} + P_{\text{net},\ell} \quad (4.58) \]
Using the fact that
\[ n_{e,t}^{n+1}kT_{e,t}^{n+1} - n_{e,t}^{n+1}kT_{e,t}^n = n_{e,t}^{n+1}\delta kT_{e,t} + n_{e,t}^{n+1}kT_{e,t}^n - n_{e,t}^{n+1}kT_{e,t}^n \]
\[ = n_{e,t}^{n+1}\delta kT_{e,t} + \delta n_{e,t}kT_{e,t}^{n+1}, \]
subtraction of (4.54) from (4.58) and multiplication by $\Delta t$ yields
\[
\frac{3}{2} n_{e,t}^{n+1}\delta kT_{e,t} - \nabla \cdot \left( \frac{5}{2} \frac{\Delta t n_{e,t}^{n+1/2}kT_{e,t}^{n+1/2}}{m_e e_n} \nabla kT_{e,t} \right) = \frac{3}{2} \delta n_{e,t}kT_{e,t}^{n+1} - \nabla \cdot \left( \frac{5}{2} \Delta t \delta T_{e,t}^{n+1/2}kT_{e,t}^{n+1/2} \right) + D_R(\delta F_{\text{energy}}) - D_R(\delta F_{\text{heat}})
\]

4.3 Solution of the Composite Systems

The synchronization of potential and temperature lead to the composite linear systems (4.50)-(4.51) and (4.60), respectively. To solve these systems, we generalize the approach described in [10] for Poisson's equation to these variable coefficient cases. This is a multilevel algorithm that iterates over the grid hierarchy in a multigrid-like fashion, taking care to enforce the required elliptic matching conditions and using high-order interpolation to preserve accuracy at coarse-fine boundaries. The algorithm can be summarized as

\[
R := f - L(u).
\]
\[
\text{While}(|R| > \epsilon |f|)
\]
\[
R := f - L(u),
\]
\[
\text{MGRelax}(\ell_{\text{max}})
\]
\[
\text{EndWhile}
\]

Procedure MGRelax($\ell$):

If ($\ell = \ell_{\text{max}}$) then
\[
R^\ell := f\ell - L^{n\ell}(u^\ell, u^{\ell-1})
\]
If ($\ell > \ell_{\text{base}}$) then
\[
u^\ell_{\text{save}} := u^\ell
\]
\[
eg^\ell - 1 := 0
\]
\[
e^\ell := \text{Smooth}(e^\ell, R^\ell, h^\ell)
\]
\[
u^\ell := u^\ell + e^\ell
\]
\[
R^{\ell - 1} := \left\{ \begin{array}{ll}
(R^\ell - L^{n\ell}(e^\ell, e^{\ell-1})) & \text{on } G^\ell_{\ell-1} \\
(f^\ell_{\ell-1} - L^{n\ell-1}(u^\ell)) & \text{on } G^u_{\ell-1}
\end{array} \right.
\]
\[
\text{MGRelax}(\ell - 1)
\]
\[
e^\ell := e^\ell + \text{Interpolate}(e^{\ell-1})
\]
\[
R^\ell := R^\ell - L^{n\ell}(e^\ell, e^{\ell-1})
\]
\[
\delta e^\ell := 0
\]
\[
\delta e^\ell := \text{Smooth}(\delta e^\ell, R^\ell, h^\ell)
\]
\[
e^\ell := e^\ell + \delta e^\ell
\]
\[
u^\ell := u^\ell_{\text{save}} + e^\ell
\]
Else
\[
\text{Smooth } L^{\ell_{\text{base}}}(e_{\text{base}}) = R_{\ell_{\text{base}}}
\]
\[
u_{\text{base}} := u_{\text{base}} + e^0
\]
EndIf
Here, \( L^L \) denotes the operator in the left hand side of (4.50)–(4.51) or (4.60), and \( f^L \) denotes the corresponding right-hand side. The operator \( L^{nf,L} \) denotes the single-level operator corresponding to \( L^L \), unmodified to account for finer level information (the \( nf \) in the superscript means “no fine”), i.e., the portion of the operator \( L^L \) included in the \( D_R \) terms is omitted. For the Smooth() function, we use red-black Gauss-Seidel relaxation, and for the Interpolate() step we use piecewise-constant interpolation with quadratic interpolation of boundary conditions as described in [10].

Our experience has been that the above multilevel algorithm is very effective in solving the temperature correction composite system (4.60). For the potential correction system (4.50)–(4.51) this algorithm may converge slowly, or even diverge, unless small timesteps are used. For this reason, we use a single cycle of the multilevel algorithm as a preconditioner in a conjugate gradient iteration, which significantly increases robustness and efficiency.

5 Software Implementation

The algorithm described here has been implemented in the Adaptive Plasma Model (APM) computer code. APM is a hybrid C++/FORTRAN code built upon an object oriented adaptive mesh refinement framework [6]. A library called AMRLib has been implemented to facilitate the development of applications using this framework. AMRLib is in turn based on the BoxLib [15] C++ class library which is designed to efficiently manage rectangular grids. The use of AMRLib and BoxLib enable APM and other applications [1] to share code for tasks such as grid manipulation, storage allocation and linear algebraic operations.

A benefit of an object-oriented approach for scientific applications is that complex data structures and operations upon them may be represented at an abstract level. This allows codes to read much more like the underlying mathematics than would be possible in FORTRAN. Major code modifications are more easily made and reuse of code is facilitated. However, we do not want these advantages to come at the cost of performance degradation. For this reason, in APM the computationally intensive routines are written in FORTRAN. The interface between C++ and FORTRAN occurs at the single grid level. Computations on a single rectangular grid can be handled easily and efficiently in FORTRAN, and represent the bulk of floating point operations. In this way we get the benefits of a modern object oriented language without any appreciable loss of efficiency.

6 Results

In this section we present numerical results which will help determine the accuracy and efficiency of the method described previously. In order to assess the accuracy of the single grid algorithm, the APM code is compared to the INDUCT [13] code on a uniform grid and the results are discussed. The accuracy of APM is analyzed by comparisons to uniform grid calculations with varying resolutions. The efficiency of the code is illustrated with timings relative to equivalent uniform grid calculations. Finally, two sample calculations are described and results are presented.

6.1 Comparison to INDUCT

The INDUCT code has been used in the semiconductor processing industry and has been benchmarked against experiment [4], [13], [14]. The governing equations used by APM are very similar to those used by INDUCT. INDUCT includes many additional features and capabilities such as neutral flow, an RF coil model coupled to the plasma, multiple ion species with variable temperature,
and complex chemistry. For this comparison, these features were not used.

A difficulty in comparing the codes is the individual coordinate systems. APM and INDUCT are both two-dimensional codes but INDUCT uses cylindrical coordinates and APM uses Cartesian coordinates. A one-dimensional comparison can be made, however, if the cylindrical case has a very large aspect ratio ($Z_{\text{scale}} \ll R_{\text{scale}}$) and the solution on the centerline is used. This can then be compared to a Cartesian calculation with suitably chosen symmetry boundary conditions. This was done and the results are shown in Figures 6.3-6.6. The simulation length was 4.75 cm, half of a 9.5 cm region with a symmetry boundary. The neutral background gas was assumed to be Argon at 60 mTorr pressure. Ionization and elastic collision data was obtained by interpolation from the same tabular data in both codes [8]. The grid spacing was 0.25 cm and the calculations were run to 500 $\mu$s at which time both codes were in steady state. The inductive power was a uniformly distributed 0.2368 watts/cm$^3$. Both codes started with constant electron and ion density profiles of $10^{11}$ cm$^{-3}$ and zero particle velocities. The initial electron temperature was 2.0 eV and the constant ion and neutral temperatures were 0.05 eV.

The results generally agree, but the differences are large enough to deserve consideration. The differences in the results can be attributed to a number of factors. The two codes use different numerical methods and discretizations. The electron and ion advection procedures are particularly dissimilar. First-order upwind differencing was used within INDUCT while second-order Godunov differencing was used within APM. The potential boundary conditions were also slightly different. The INDUCT calculation used a cell-centered, boundary while the APM boundaries are edge centered. Considering these differences, we consider the agreement between the codes to be good.
Figure 6.4: Electron density profiles for APM and INDUCT test case.

Figure 6.5: Ion flux profiles for APM and INDUCT test case.
6.2 Accuracy Using Locally Refined Grids

Consider a locally refined grid calculation with cell sizes $h_1, h_2, ..., h_n$ where $h_n$ is the cell size corresponding to the finest cells and $h_1$ corresponding to the coarsest cells. Such a calculation should be no more accurate than a uniform grid calculation with cell size $h_n$. In fact, the degree to which the locally refined calculation matches the fine, uniform grid calculation is a measure of its accuracy.

Figure 6.7 shows a comparison of potential profiles using a locally refined grid, a fine uniform grid and a coarse uniform grid. Symmetry boundary conditions on top and bottom restrict the solution variation to the x-direction. Figures 6.8 - 6.10 shows the ion flux, electron density and electron temperature, respectively, for this same calculation. The fine grid spacing corresponds to the finest cells in the refined grid and the coarse grid spacing corresponds to the coarsest cells. The refined grid uses 3 levels. Level 1 is generated by refining the half of level 0 near the containing wall by a factor of 2. Similarly, the half of level 1 near the wall is refined by a factor of 2. This grid structure is shown in Figure 6.11. The agreement in the fine portion of the locally refined grid is excellent while the agreement in the coarse region is much better than the equivalent coarse grid.

The accuracy of the locally refined calculation depends critically on where the refined grids are used. The high accuracy of this calculation is due in large part to the placement of the fine grids near the boundary. As expected this region has the most variation in potential, density and flux. Consequently, smaller cells are required to resolve this variation. Conversely, the use of coarse grids has little effect on the quasineutral region in the center of the plasma. In this case the correct placement for the grids was clear, but in a more complex calculation this may not be the case. We are continuing to investigate the relationship between grid placement and accuracy. Most of our work to date has used heuristically placed grids, but we have had some success basing the grid
Figure 6.7: Potential profile for locally refined, fine and coarse grid calculations.

Figure 6.8: Ion flux profile for locally refined, fine and coarse grid calculations.
Figure 6.9: Electron density profile for locally refined, fine and coarse grid calculations.

Figure 6.10: Electron temperature profile for locally refined, fine and coarse grid calculations.
6.3 Computational Efficiency

In order for a local refinement strategy to make sense, refined calculations need to show some computational savings. Figure 6.12 shows the relative CPU times for uniform grid calculations compared to locally refined calculations. The model problem for these timings was a square grid with symmetry boundaries on the top and right side and grounded conductors on the other sides. At each level, the outermost 3 cells along the physical boundary were chosen for refinement. These grids are shown in Figure 6.13.

Results are shown for refinement ratios of 2 and 4. In each case calculations were performed with a refined grid and a uniform grid. The uniform grid cell size corresponded with the finest cells in the refined grid. Each pair of calculations was run to the same simulation time and the ratio of CPU times was calculated. The final time for each pair of calculations was the same but different pairs used different final times. Our experience has been that the CPU time per iteration does not vary much during a calculation, so these timings should be representative.

The finest of these calculations cannot be run to steady state in a reasonable amount of time. Even with the speedup shown with local refinement, very high resolution calculations are still very expensive for these static grids. Ultimately we will be changing the grid structure in time to adapt to the transient solution. Great savings can be made by gradually refining the grid as the solution approaches steady state. We are investigating this approach, which was used for the sample calculations described below.

The speedup is a strong function of refinement ratio. For the model problem, 4 refined levels were required to obtain an improvement over the single fine grid case using a refinement ratio of 2. When a refinement ratio of 4 was used, only 2 refined levels were required to get a significant speedup. These trends depend on a number of factors. Chief among these are the time spent performing composite solves and the fraction of the domain covered by fine cells. For this model problem, the number of cells at the fine levels was rather large compared to the coarse levels.
Figure 6.12: Ratio of CPU times for uniform grid calculations to locally refined grid calculations.

is required if one wants to refine along an entire edge. If, for example, a corner was recursively
refined we would expect an ever larger speedup for the locally refined grid.

The dependence of time spent doing composite solves on the grid structure is hard to quantify. We do know that the number of composite solves grows quickly with the number of levels. If \( m \)
refined levels are used and each is refined by a factor of \( n \) in time and space, this will require \( n^{m-1} \)
composite solves per coarse timestep. This fact and the speedup results shown in Figure 6.12 both
show the need for higher refinement ratios.

We are also investigating subcycling in time. In some cases, we have noticed a \( \Delta x^2 \) (rather
than \( \Delta x \)) dependence on the Courant stable timestep as we refine our grid. If we can base the
level timestep on its own level velocities and grid spacing rather than on the composite grid, we
can substantially reduce the number of composite solves required.

It is worth noting that runtime is not the only benefit of locally refined grids. The memory
requirements are much less severe for refined grids, provided the finest levels do not make up a
large fraction of the domain. At some level of refinement, it is no longer feasible to compare highly
refined grids to uniform equivalent grids because the latter cannot fit within physical memory.

6.4 Sample Calculations

In this section we show two sample calculations. The first case is a simple two level calculation on a
rectangular grid. The grid structure is shown in Figure 6.14 and the refinement ratio is 4. The top
and right edges are symmetry boundaries and the left and bottom edges are grounded conductors.
The long edge is 16 cm while the short edge is 8 cm. The calculation time is 50 \( \mu \)s. The initial
conditions were constant ion and electron densities of \( 10^{11} \text{ cm}^{-3} \), zero fluxes and potential, and 2.0
eV electron temperature. The electron density and potential are plotted as two-dimensional bar charts (Figures 6.16-6.17), with the data points corresponding to the bar centers, in order to better see the underlying grid structure. Adjacent bars with the same height are actually the same cell. On the fine level, each bar corresponds to a single cell.

The second case is a more computationally intensive three level case with refinement ratios of 2 (Figure 6.15). The same symmetries as in the previous case are used, but the domain is an 8 cm square. Contour plots are shown for potential, ion density, electron density and temperature along with the grid outlines (Figure 6.18 - 6.21). The ion flux is drawn as a vector plot (Figure 6.22). The results are shown at 20 µs, and the same initial conditions were used as in the previous calculation. The local refinement was adaptive in this case with the first refined level being added at 14 µs and the second at 15 µs.

Acknowledgement

We wish to thank Peter Vitello for his many helpful discussions regarding plasma modeling and his assistance in comparing our code with INDUCT.
Figure 6.14: Geometry for first sample calculation.

Figure 6.15: Geometry for second sample calculation.
Figure 6.16: Potential profile for first sample APM calculation.

Figure 6.17: Electron density profile for first sample APM calculation.
Figure 6.18: Potential contours for the second sample APM calculation. Contours range from 1.09 to 15.85 volts.

Figure 6.19: Electron density contours for the second sample APM calculation. Contours range from \(4.08 \times 10^9\) to \(1.46 \times 10^{12}\) cm\(^{-3}\).
Figure 6.20: Ion density contours for the second sample APM calculation. Contours range from $4.15 \times 10^9$ to $1.46 \times 10^{12}$ cm$^{-3}$.

Figure 6.21: Electron temperature contours for the second sample APM calculation. Contours range from 2.12 to 2.74 eV.
References


coupled plasma with comparison to experimental spatial profiles. *Journal of Applied Physics*,
80, September 1996.


Laboratory, Livermore, California, April 1993.

1974.

Physics 56(12) 6068, 1972.


[10] D. F. Martin and K. L. Cartwright. Solving Poisson’s equation using adaptive mesh refine-
ment. Technical Report UCB/ERL M96/66 Electronics Research Laboratory Memorandum


