ALGORITHM OVERVIEW

This document synthetizes the general methodology behind the code developed to simulate the Direct-Fusion Drive for low-power, experimental scenarios. As expected, the designed MATLAB code simulates the cold plasma dynamics and propulsive properties of a Direct-Fusion Drive (DFD) experimental system running at low powers. It takes into account multiple physical, thermodynamic, and computational factors, as we'll see. The code is structured into several key sections: initialization, setup of dynamical variables, main simulation loop, and complementary functions for computation and visualization.

0.1 Initialization and Setup

The simulation begins by clearing the MATLAB environment and initializing a runtime counter using 'tic'. Physical constants and parameters relevant to the DFD system are defined, including the Boltzmann constant (k_B), ionization energy of deuterium (E_{ion}), mass of deuterium (m_D), and electrons (m_e), system dimensions (L), mass flow rate ($\dot{m}$), specific heat ratio (γ), input power (P_{inp}), and other constants.

The spatial domain is defined with 3000 grid points (N_z), and the spatial step size (dz), and normalized spatial domain, (Z), are computed. Error parameters are initialized to control the simulation accuracy.

Normalization factors are calculated to non-dimensionalize the variables. These factors include area (A^*), temperature (T^*), velocity (v^*), density (n^*), time (t^*), ionization rate coefficient (R^*), plasma heating (AQ^*), power (P^*), and force (F^*) normalization factors. These normalizations simplify the equations and enhance numerical stability.

The area (A) and sigma (σ) function of the system are computed by loading and interpolating the area introduced in section 3.1.3. The sigma function represents the logarithmic derivative of the area, which is present within in our plasma equations model, as shown in section 3.1.2.

The energy source function (AQ^*) for plasma heating is defined using a Gaussian profile. This profile models the distribution of heating power along the normalized spatial domain. The ionization rate coefficient ($Rion|_0$) is also precomputed for a range of electron temperatures to expedite the simulation. As seen in section 3.1.5, this coefficient determines the rate at which neutral particles ionize into plasma, affecting the overall dynamics and performance of the DFD.

Afterwards, the dynamic variables such as neutral velocity (nn), thrust (F), propellant efficiency (η), utilization efficiency (η_u), specific impulse (I_{sp}), Mach number (Ma), and energy loss (E_{lost}) are initialized. The code provides an option to load previous simulation data if available, allowing for continuity between simulation runs. If no previous data exists, default initial conditions are used to initialize the plasma properties.

0.2 Main Simulation Loop and Data Visualization

The core of the simulation is the main loop, which iterates until the error variable ($errVar$) falls below a specified threshold ($MaxError$). The loop performs several key operations:

- Intermediate Plotting: Every 1000 iterations, intermediate results are plotted to visualize the current state of the simulation;
- Time Step Calculation: The time step (dt) is updated based on the Courant-Friedrichs-Lewy (CFL_{lim}) condition, presented in section 3.2.3;
- Flux Computations: Fluxes of the main variables are computed using the Local Lax-Friedrichs (LLF) scheme, previously introduced in section 3.2.1.1;

- Variable Updates: The primary variables are updated following the Runge-Kutta method (RK-4), seen in section 3.2.2. Boundary conditions are applied to maintain physical consistency at the edges of the spatial domain;
- Dynamical Properties Calculation: The plasma dynamical variables such as electron density, ion velocity, electron temperature, neutral density, and the sonic speed are recalculated;
- Performance Metrics: Key performance metrics, including thrust, specific impulse, propellant efficiency, and utilization efficiency, are computed;
- Error Calculation: The mass flow error is evaluated to control the convergence of the simulation. If the error is within the acceptable range, the loop continues; otherwise, adjustments are made to improve accuracy.

After the main loop concludes, the total energy lost due to ionization is calculated, E_{lost}, providing insight into the energy efficiency of the DFD system. The final results are then visualized, and the simulation runtime is displayed to the user.

The simulation data is saved for future use, allowing for continuity between runs and faster convergence. This includes interpolating final values to a standard grid for consistent data analysis.

0.3 Complementary Functions

The Direct-Fusion Drive (DFD) simulation algorithm is supported by several complementary functions, each serving a specific role in the computational process:

- Area and sigma function:
  The 'Asig' function calculates the area of the system and the sigma function, which is the logarithmic derivative of the area, necessary for the flux computations in plasma dynamics. It takes the spatial domain grid, area data, and sigma data as inputs and outputs the interpolated area and sigma values over the spatial domain. The process involves loading predefined area and sigma data, interpolating these values for smooth and continuous representations, and calculating the sigma function as the logarithmic derivative of the interpolated area;
- Computation of dynamical variables:
  The 'NewVar' function recalculates secondary variables and dynamical properties from the primary simulation variables. Inputs include main variables such as densities, velocities, and energy, while outputs consist of recalculated secondary variables like electron density, ion velocity, electron temperature, neutral density, and sonic speed. This function computes the electron density from ion and neutral densities, calculates electron temperature and ion velocity based on energy and momentum equations, updates neutral density considering ionization rate and mass conservation, and determines the sonic speed for use in the main code;
- Ionization rate coefficient function:
  The 'fRion' function calculates the ionization rate coefficient over a range of electron temperatures, an essential element in determining the rate at which neutral particles ionize into plasma. It takes a range of electron temperatures as input and outputs the ionization rate coefficient as a function of electron temperature. The process involves using physical models or empirical formulas to compute the ionization rate coefficient for each temperature value and storing these coefficients in a table for efficient lookup during the main simulation loop, thereby reducing computational overhead;

- Data visualization function:
  The 'DataVis' function handles the visualization of simulation data, producing plots of the plasma dynamical variables. It takes the current state of main and secondary variables and performance metrics as input and generates plots and numerical results displayed on the screen.

- Local Lax-Friedrich (LLF) function:
  The LLF function implements the Local Lax-Friedrichs scheme for flux computations. Inputs include the current state of main variables, fluxes, and system properties, with the output being updated fluxes for the main variables. The process involves computing fluxes using the Local Lax-Friedrichs scheme by averaging values and incorporating dissipative terms for numerical stability, calculating temporal derivatives for densities, velocities, and energy, and ensuring adherence to the Courant-Friedrichs-Lewy (CFL) condition for stability.

The modular approach of the code by working out these aspects of the code on distinct functions enhances the performance of the code and allows the user to effectively debug different sections of the program.