On Preconditioning for the Two-Field Neutral Model

J. Christopher*[1], A. Davis†[1], D. Ghosh ‡[2], M. Dorf§[2], M. Dorr¶[2], L. Ricketson ‖[2], and X. Gao **[1]
[1]*Computational Fluid Dynamics and Propulsion Laboratory , Colorado State University, Fort Collins, CO, USA*
[2]*Center for Applied Scientific Computing , Lawrence Livermore National Laboratory, Livermore, CA, USA*

Achieving high performance in accuracy and efficiency for the numerical modeling of edge plasma is challenging because of the extreme anisotropy of the transport equations for complex physical processes. The multiscale nature calls for implicit-explicit (IMEX) or fully-implicit time integration methods for the sake of stability and efficiency. Nevertheless, the implicitness introduces nonlinear and linear solvers in the solution process whose efficiency usually depends on preconditioners. Therefore, the present study aims to explore preconditioning for nonlinear terms to attain an efficient solver for the spatial discretization matrices and thus allow for large time steps. A new preconditioner is derived and implemented for a two-field fluid neutral model. Its application to solve the model with a slab configuration that represents a 2D section in the longitudinal-radial plane of the edge geometry demonstrates expected solution accuracy but performance needs further improvement.

I. Nomenclature

I	implicit term	T_g	neutral species temperature
E	explicit term	$V_{\|i}$	parallel ion velocity
g	neutral gas	$V_{\|g}$	parallel neutral velocity
i	ion	A	a dummy scalar variable
iz	ionization	$\boldsymbol{b}$	magnetic field unit vector
r	recombination	$\langle\sigma v\rangle_{cx}$	charge exchange cross section
ν_{iz}	ionization frequency	$\langle\sigma v\rangle_{iz}$	ionization cross section
ν_r	recombination frequency	n_i	number density of ion
ν_{cx}	charge-exchange frequency	D	normalized diffusivity
m_g	neutral species mass	μ	normalized viscosity

II. Introduction

Numerical modeling of edge plasmadynamics requires a hierarchy of computational tools and physics models depending on the level of complexity to be considered. Challenges in edge plasma modeling are presented by the multiscale nature due to the inhomogeneous magnetic field driving complex physics through magnetic mirroring, ballooning, toroidal mode coupling, and curvature drift as well. The integration of physical processes, such as those describing the edge localized modes including magneto-fluid dynamics, turbulence, neutral transport, and plasma-wall interactions, would make the system of partial differential equations (PDEs) that govern the rich physics daunting to solve. Effective and efficient numerical strategies need to be developed for understanding complex edge plasma physics with non-equilibrium thermodynamics and non-Maxwellian distributions. The present study is focused on the development of a preconditioner for nonlinear and linear solvers resulting from the use of implicit-explicit (IMEX) or fully implicit time integration methods, which are used to overcome severe time step constraints imposed by the stiff PDEs when using explicit methods.

*PhD, Corr. Author, AIAA member, Joshua.Christopher@colostate.edu
†PhD Student, andrew.davis@colostate.edu
‡Staff Scientist, ghosh5@llnl.gov
§Staff Scientist, dorf1@llnl.gov
¶Senior Scientist, dorr1@llnl.gov
‖Staff Scientist, ricketson1@llnl.gov
**Associate Professor, AIAA senior member, gao@colostate.edu

The computational framework, COGENT (COntinuum Gyrokinetic Edge New Technology) developed at Lawrence Livermore National Laboratory [2], provides the base for the development and testing of preconditioners in the present study. The underlying physics model in COGENT is a 4D/5D, Eulerian, gyrokinetic Vlasov-Poisson system with an imposed electromagnetic field. The computational domain spans from the core region, across the magnetic separatrix, and into the scrape-off layer. The structure of the edge is illustrated by Fig. 1. The spatial discretization scheme is based on finite volume methods (FVMs) and, by design, the discretization schemes are fourth-order accurate in space and time. Due to the enormous disparity in time scales associated with PDEs governing the complex edge plasmadynamics, fully-implicit or implicit-explicit (IMEX) time marching methods are preferred to fully-explicit ones for numerical stability and solution efficiency.

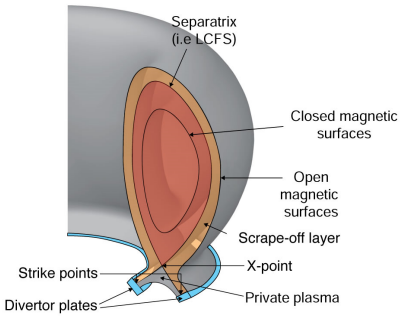

Fig. 1 Illustration of the edge geometry: the edge region is defined from the outer wall, across the separatrix, then to the last closed magnetic field line/surface of the core region [1].

Nevertheless, numerical challenges present with the implicitness of time integration. Fully implicit or IMEX methods require solving a nonlinear system at each time step, introducing significant computational cost for high-dimensional complex problems and nontrivial parallelization difficulties. More than often, slow convergence or even divergence occurs when the nonlinear system, or the linear system resulting from the linearization of this nonlinear system, is not well preconditioned. Therefore, efficient and robust nonlinear and linear solvers are needed, and designing good preconditioners is the essential key. As a starting point, this study is focused on the derivation, implementation, and test of a preconditioner based on the analytical Jacobians (e.g., source Jacobian, flux Jacobian, or collectively residual Jacobian, depending on terms to be treated implicitly). The preconditioner performance is demonstrated using the two-field neutral model for a 2D slab geometry and compared to that of the existing preconditioner.

III. Mathematical Equations of the Two-field Neutral Model

We focus on the fluid neutral model, specifically, the two-field neutral model in physical space [3], described by

$$\frac{\partial \rho_g}{\partial t} + \vec{\nabla}\cdot\left(V_{\|g}\,\mathbf{b}\,\rho_g\right) - \vec{\nabla}\cdot\left(D\,\vec{\nabla}_\perp\left(\rho_g\,T_g\right)\right) = \left(\nu_r - \nu_{iz}\right)\rho_g\,, \tag{1}$$

$$\begin{aligned}\frac{\partial \rho_g V_{\|g}}{\partial t} &+ \vec{\nabla}\cdot\left(\left[V_{\|g}\mathbf{b} - D\rho_g^{-1}\vec{\nabla}_\perp\left(\rho_g T_g\right)\right]\rho_g V_{\|g}\right) - \vec{\nabla}\cdot\left(\mu\vec{\nabla}V_{\|g}\right)\\ &= -\vec{\nabla}_\|\left(\frac{\rho_g}{m_g}T_g\right) + \rho_g\nu_{cx}\left(V_{\|i} - V_{\|g}\right) - \nu_{iz}\rho_g V_{\|g} + \nu_r\rho_g V_{\|i}\,,\end{aligned} \tag{2}$$

where $\vec{\nabla}_\perp A = \vec{\nabla}A - \mathbf{b}(\mathbf{b}\cdot\vec{\nabla})A$ and the $_{\|g}$ direction is defined as the direction parallel to the field line. The normalized diffusivity (D), viscosity (μ), charge-exchange frequency (ν_{cx}), and ionization frequency (ν_{iz}) coefficients are given by

$$D = m_g^{-1}\nu_{cx}^{-1}, \qquad \mu = \frac{\rho_g T_g}{m_g \nu_{cx}}, \qquad \nu_{cx} \equiv \langle\sigma v\rangle_{cx} n_i, \qquad \nu_{iz} \equiv \langle\sigma v\rangle_{cx} n_i\,. \tag{3}$$

The ion number density is given by n_i (number of particles/volume). Recombination occurs when positive ions in the plasma capture a free electron and form neutral atoms, thus, the recombination frequency, ν_r, describes the rate of this recombination process. In $\langle\sigma v\rangle$, v is the electron velocity and is a function of electron energy, $v = \sqrt{2\langle E\rangle/m}$ where $\langle E\rangle = k_b T_e$, k_b is the Boltzmann constant, T_e is the electron temperature, and m is the electron mass.

Rewrite Eqns.(1)-(2) in vector notation as

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla}\cdot\left(\vec{\mathbf{E}} - \vec{\mathbf{F}}\right) = \mathbf{S}(\mathbf{U})\,, \tag{4}$$

where $\mathbf{U}$ is the solution vector, $\mathbf{U} = \left[\rho_g, \rho_g V_{\|g}\right]^{\mathrm{T}}$, $\vec{\mathbf{E}}$ is the convective flux dyad

$$\vec{\mathbf{E}} = \begin{bmatrix} V_{\|g}\,\mathbf{b}\,\rho_g \\ \left[V_{\|g}\mathbf{b} - D\rho_g^{-1}\vec{\nabla}_\perp\left(\rho_g T_g\right)\right]\rho_g V_{\|g}\end{bmatrix}, \tag{5}$$

and $\vec{\mathbf{F}}$ is the diffusive flux dyad

$$\vec{\mathbf{F}} = \begin{bmatrix} D\,\vec{\nabla}_\perp\left(\rho_g\, T_g\right) \\ \mu\vec{\nabla}V_{\|g} \end{bmatrix}, \tag{6}$$

and $\mathbf{S}$ is the source vector in physical space

$$\mathbf{S} = \begin{bmatrix} \left(v_r - v_{iz}\right)\rho_g \\ -\vec{\nabla}_\| \left(\frac{\rho_g}{m_g} T_g\right) + \rho_g \nu_{cx}\left(V_{\|i} - V_{\|g}\right) - v_{iz}\rho_g V_{\|g} + v_r \rho_g V_{\|i} \end{bmatrix}. \tag{7}$$

Note that for generality, we call the flux dyads even though the current model is simplified with one flux vector (in the direction parallel to the field line).

COGENT employs mapped grids to accommodate complex geometry. The computational space is Cartesian and facilitates the use of adaptive mesh refinement. In this paper, we present the mathematical models and Jacobians in both physical and computational spaces for convenience.

Assume a smooth mapping from some abstract coordinate space into physical space $\vec{x} = \vec{x}(\vec{\xi})$, $\vec{x} : [0,1]^D \to \mathbb{R}^D$, where $\vec{x}$ denotes the physical space, eg. (x, y, z) for 3D in space, and $\vec{\xi}$ represents the computational space, eg. (ξ, η, ζ) for 3D in space. The divergence of a vector field, $\vec{\mathcal{F}}$, in physical space can be expressed in computational space as

$$\vec{\nabla}_x \cdot \vec{\mathcal{F}} = \frac{1}{J}\vec{\nabla}_\xi \cdot \left(\mathrm{N}^{\mathrm{T}}\vec{\mathcal{F}}\right), \tag{8}$$

where J is the grid metrics Jacobian and N^{T} contains grid metrics. $\vec{\nabla}_x$ and $\vec{\nabla}_\xi$ are the vector differential operator in the physical and the computational space, respectively. Assuming the coordinates are time invariant, we obtain

$$J \equiv \det(\vec{\nabla}_\xi \vec{x}), \quad \mathrm{N}^{\mathrm{T}} = J\,\vec{\nabla}_x\vec{\xi}, \quad \mathrm{N} = J\,(\vec{\nabla}_x\vec{\xi})^{\mathrm{T}}, \quad \text{and } \mathrm{N}^{\mathrm{T}}{}_{d,s} \equiv \det\left((\vec{\nabla}_\xi \vec{x})^{\mathrm{T}}(d|\boldsymbol{e}^s)\right),$$

where $\mathrm{A}(d|\boldsymbol{e}^s)$ denotes a modification of matrix A by replacing row d with vector $\boldsymbol{e}^s$.

In computational space, $\vec{\xi}$, Eq. (4) is transformed using grid metrics as

$$\frac{\partial(J\mathbf{U})}{\partial t} + \vec{\nabla}_\xi \cdot \left(\mathrm{N}^{\mathrm{T}}\left(\vec{\boldsymbol{\mathcal{E}}} - \vec{\boldsymbol{\mathcal{F}}}\right)\right) = J\boldsymbol{\mathcal{S}}. \tag{9}$$

Where $J\mathbf{U}$ is the solution vector $J\mathbf{U} = \left[J\rho_g, J\rho_g V_{\|g}\right]^{\mathrm{T}}$. In the divergence term, the hyperbolic flux dyad would remain exactly the same as that in physical space if it was strictly convective or hyperbolic, but here it contains the gradient of $\rho_g T_g$, which requires grid metrics for the transformation. Therefore, we use $\vec{\boldsymbol{\mathcal{E}}}$ to distinguish it from $\vec{\mathbf{E}}$. Similarly, the source vector includes a derivative term, and $\boldsymbol{\mathcal{S}}$ replaces $\mathbf{S}$. In general, the viscous (elliptic) flux, $\vec{\boldsymbol{\mathcal{F}}}$, is different from that in physical space because it always involves derivatives. The flux dyads and source vector in computational space are given by

$$\vec{\boldsymbol{\mathcal{E}}} = \begin{bmatrix} V_{\|g}\mathbf{b}\rho_g \\ \left[V_{\|g}\mathbf{b} - D\rho_g^{-1}\frac{\mathrm{N}}{J}\vec{\nabla}_{\xi\perp}\left(\rho_g T_g\right)\right]\rho_g V_{\|g} \end{bmatrix}, \quad \vec{\boldsymbol{\mathcal{F}}} = \begin{bmatrix} \vec{Q} \\ \vec{\vec{\mathcal{T}}} \end{bmatrix}, \tag{10}$$

$$\boldsymbol{\mathcal{S}} = \begin{bmatrix} \left(v_r - v_{iz}\right)\rho_g \\ -\frac{\mathrm{N}}{J}\vec{\nabla}_{\xi\|}\left(\frac{\rho_g}{m_g}T_g\right) + \rho_g\nu_{cx}\left(V_{\|i} - V_{\|g}\right) - v_{iz}\rho_g V_{\|g} + v_r\rho_g V_{\|i} \end{bmatrix}. \tag{11}$$

For clarity, we use $\vec{\nabla}_\xi$ to denote the operator in computational space, while either $\vec{\nabla}_x$ or $\vec{\nabla}$ represents the operator in physical space. In $\vec{\boldsymbol{\mathcal{F}}}$ of Eq. (10), $\vec{Q}$ is the mapped heat flux vector and $\vec{\vec{\mathcal{T}}}$ is the mapped stress tensor. Here, note that the mapped stress tensor is a simplified version.

In general, gradients of scalars (ϕ) or vectors $(\vec{\phi})$ in physical space are derived using the chain rule,

$$\vec{\nabla}_x\phi = \frac{\mathrm{N}}{J}\vec{\nabla}_\xi\phi \quad \text{and} \quad \vec{\nabla}_x\vec{\phi} = (\vec{\nabla}_\xi\vec{\phi})\frac{\mathrm{N}^{\mathrm{T}}}{J}. \tag{12}$$

Therefore, the mapped strain tensor is given by

$$\vec{\nabla}_x \vec{u} = (\vec{\nabla}_\xi \vec{u})\ \frac{\mathrm{N}^\mathrm{T}}{J}. \tag{13}$$

The heat flux is modeled using Fourier's law and given by

$$\vec{Q} = -D\frac{\mathrm{N}}{J}\vec{\nabla}_\xi(\rho_g T_g)\,. \tag{14}$$

The mapped stress tensor (in the "full version" which is kept here for future reference) is defined by

$$\vec{\vec{\mathcal{T}}} = 2\mu\left(\vec{\vec{S}} - \frac{1}{3}J^{-1}\vec{\vec{I}}\,\vec{\nabla}_\xi \cdot \left(\mathrm{N}^\mathrm{T}\vec{u}\right)\right), \tag{15}$$

where the viscosity is μ and the strain rate tensor, $\vec{\vec{S}}$, is in general defined by

$$\vec{\vec{S}} = \frac{1}{2}\left((\vec{\nabla}_\xi \vec{u})\ \frac{\mathrm{N}^\mathrm{T}}{J} + \left((\vec{\nabla}_\xi \vec{u})\ \frac{\mathrm{N}^\mathrm{T}}{J}\right)^\mathrm{T}\right). \tag{16}$$

However, in the current simplified model, the mapped stress tensor is evaluated by the following with only the parallel component of $\vec{u}$ (e.g., $V_{\|i}, V_{\|g}$)

$$\vec{\vec{\mathcal{T}}} = 2\mu\vec{\vec{S}}, \quad \text{with} \quad \vec{\vec{S}} = \frac{1}{2}(\vec{\nabla}_\xi \vec{u})\ \frac{\mathrm{N}^\mathrm{T}}{J}\,, \tag{17}$$

where factors 2 and 1/2 are kept for convenience in future extension in the code implementation.

IV. The Semi-Discrete Form from the Application of Finite Volume Methods

Applying the finite volume method on mapped grids, Eq. (9) then becomes

$$\frac{\partial}{\partial t}\int_{V_{\boldsymbol{i}}} J\mathbf{U}\,\mathrm{d}\vec{\xi} + \int_{V_{\boldsymbol{i}}} \vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T}(\vec{\boldsymbol{\mathcal{E}}} - \vec{\boldsymbol{\mathcal{F}}}))\,\mathrm{d}\vec{\xi} = \int_{V_{\boldsymbol{i}}} J\boldsymbol{\mathcal{S}}\,\mathrm{d}\vec{\xi}\,. \tag{18}$$

The semi-discrete form on mapped grids is

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle J\mathbf{U}\rangle_{\boldsymbol{i}} = -\frac{1}{\Delta\xi_d}\sum_{d=1}^{D}\left(\left(\langle \mathrm{N}^\mathrm{T}{}_d\vec{\boldsymbol{\mathcal{E}}}\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathrm{N}^\mathrm{T}{}_d\vec{\boldsymbol{\mathcal{E}}}\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right) - \left(\langle \mathrm{N}^\mathrm{T}{}_d\vec{\boldsymbol{\mathcal{F}}}\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathrm{N}^\mathrm{T}{}_d\vec{\boldsymbol{\mathcal{F}}}\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right)\right) + \langle J\boldsymbol{\mathcal{S}}\rangle_{\boldsymbol{i}}\,, \tag{19}$$

where the subscript d denotes the d^{th} row of N^T, $\langle\vec{\boldsymbol{\mathcal{E}}}\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ and $\langle\vec{\boldsymbol{\mathcal{F}}}\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ are the mapped viscous flux dyads at cell faces. Angle bracket, $\langle\cdot\rangle$, represents either cell-averaged or face-averaged quantity.

V. Implicitness and Preconditioning

In the study, the preconditioners for both the IMEX and the fully implicit methods are derived for Eq. (19). The existing approximate preconditioner is briefly reviewed first. Then, the new preconditioner based on the analytical Jacobians for the source and fluxes is derived in computational space. The new preconditioner is implemented in COGENT and its performance is evaluated and compared to the existing preconditioner.

A. The Implicit-Explicit (IMEX) Method

Both the viscous and source terms are treated implicitly while the inviscid is explicitly evolved. Denote $\vec{\mathcal{F}}_I$ and $\vec{\mathcal{F}}_E$ as the implicit and explicit terms, respectively

$$\vec{\mathcal{F}}_I = \mathrm{N}^\mathrm{T}\vec{\boldsymbol{\mathcal{F}}} + J\boldsymbol{\mathcal{S}}\,, \tag{20}$$

$$\vec{\mathcal{F}}_E = -\mathrm{N}^\mathrm{T}\vec{\boldsymbol{\mathcal{E}}}\,, \tag{21}$$

where subscripts $_I$ and $_E$ stand for implicit and explicit, respectively. Following Ghosh [4] and the COGENT notation system, we denote the solution vector by $\mathbf{y} \equiv J\mathbf{U}$; that is $[y_1, y_2] = \left[J\rho_g, J\rho_g V_{\|g}\right]^{\mathrm{T}}$. Integrate Eq. (19) in time using an additive Runge-Kutta (ARK) method. For example, we express the nonlinear function resulting from the IMEX time integration in form of

$$\boldsymbol{\mathcal{G}}\left(\mathbf{y}\right) \equiv \alpha\mathbf{y} - \vec{\boldsymbol{\mathcal{F}}}_I\left(\mathbf{y}\right) - \mathbf{r} = 0\,, \tag{22}$$

where

$$\alpha = \frac{1}{\Delta t \tilde{a}_{ll}}, \qquad \mathbf{r} = \frac{1}{\Delta t \tilde{a}_{ll}}\left(\mathbf{y}^n + \Delta t \sum_{j=1}^{l-1}\left\{\tilde{a}_{lj}\vec{\boldsymbol{\mathcal{F}}}_E\left(\mathbf{y}^j\right) + \tilde{a}_{lj}\vec{\boldsymbol{\mathcal{F}}}_I\left(\mathbf{y}^j\right)\right\}\right). \tag{23}$$

The α's are the coefficients from Butcher table associated with ARK methods. The term $\mathbf{r}$ is a function of old (or known) value of $\mathbf{y}$.

Eq. (22) is solved using the Newton's method and the brief steps are provided as follows:

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \left[\mathcal{J}\left(\mathbf{y}^k\right)\right]^{-1}\boldsymbol{\mathcal{G}}\left(\mathbf{y}^k\right), \tag{24}$$

where the superscript k is the Newton iteration index and the Jacobian is given by

$$\mathcal{J}\left(\mathbf{y}\right) = \frac{d\boldsymbol{\mathcal{G}}\left(\mathbf{y}\right)}{d\mathbf{y}}\,. \tag{25}$$

Linear Solve Eq. (24) is a linear system, a Δ-form that solves for $\Delta\mathbf{y} \equiv \mathbf{y}^{k+1} - \mathbf{y}^k$ using a preconditioned generalized minimum residual (GMRES) methods for solution efficiency as

$$\mathcal{J}\left(\mathbf{y}^k\right)\Delta\mathbf{y} = -\boldsymbol{\mathcal{G}}\left(\mathbf{y}^k\right)\,. \tag{26}$$

Preconditioning The right preconditioned GMRES algorithm expresses Eq. (26) as

$$\mathcal{J}\left(\mathbf{y}^k\right)\mathcal{P}\left(\mathbf{y}^k\right)\mathcal{P}\left(\mathbf{y}^k\right)^{-1}\Delta\mathbf{y} = -\boldsymbol{\mathcal{G}}\left(\mathbf{y}^k\right), \tag{27}$$

where $\mathcal{P}\left(\mathbf{y}^k\right) \approx \mathcal{J}\left(\mathbf{y}^k\right)$ is the preconditioning matrix. This requires the preconditioning matrix $\mathcal{P}$ evaluated at the current Newton iteration solution $\mathbf{y}^k \equiv J\mathbf{U}^{i,k}$. However, to reduce the computational expense, the preconditioning matrix is updated every stage (before solving for that stage) with the initial guess for the stage solution $J\mathbf{U}^{i,0}$. Thus, the following system is solved:

$$\mathcal{J}\left(\mathbf{y}^k\right)\mathcal{P}\left(\mathbf{y}^0\right)\mathcal{P}\left(\mathbf{y}^0\right)^{-1}\Delta\mathbf{y} = -\boldsymbol{\mathcal{G}}\left(\mathbf{y}^k\right), \tag{28}$$

instead of Eq. (27), with the assumption that $\mathcal{P}\left(\mathbf{y}^0\right) \approx \mathcal{P}\left(\mathbf{y}^k\right)$.

An existing preconditioner assumes decoupling between Eqns. (1) and (2). That is, we "freeze" density in the parallel momentum equation. The preconditioner to $d\boldsymbol{\mathcal{G}}\left(\mathbf{y}\right)/d\mathbf{y}$ can therefore be taken as

$$\mathcal{P}(\mathbf{y}) = d\boldsymbol{\mathcal{G}}\left(\mathbf{y}\right)/d\mathbf{y} \tag{29}$$

$$= \alpha\mathbf{I} - \frac{d\vec{\boldsymbol{\mathcal{F}}}_I}{d\mathbf{y}}\,, \tag{30}$$

$$\mathcal{P}(\mathbf{y}) = -\begin{bmatrix} J\nabla\cdot\left(D\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) - (v_r - v_{iz} - \alpha)\circ & 0 \\ 0 & J\nabla\cdot\left(\eta\nabla(J^{-1}y_1^{-1}\circ)\right) + (v_{cx} + v_{iz} + \alpha)\circ \end{bmatrix}. \tag{31}$$

The "$\circ$" symbol indicates the "operator action" of the preconditioner. That is,

$$\mathcal{P}_{11}(\mathbf{y})[\phi] = J\nabla\cdot\left(D\nabla_\perp\left(\frac{T_g}{J}\phi\right)\right) - (v_r - v_{iz} - \alpha)\phi\,. \tag{32}$$

Let

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \bar{z}_1 J T_g^{-1} \\ \bar{z}_2 J \rho_g \end{bmatrix}. \tag{33}$$

The vector $\mathbf{z}$ is applied to the operator action of Eq. (31)

$$\mathcal{P}(\mathbf{y})[\mathbf{z}] = -\begin{bmatrix} J\nabla\cdot(D\nabla_\perp(\bar{z}_1)) - (v_r - v_{iz} - \alpha)\bar{z}_1 J T_g^{-1} & 0 \\ 0 & J\nabla\cdot(\eta\nabla(\bar{z}_2)) + (v_{cx} + v_{iz} + \alpha)J\rho_g\bar{z}_2 \end{bmatrix}. \tag{34}$$

Reformulate as a linear system and solve the preconditioner system of $\mathcal{P}[\mathbf{z}] = \mathcal{G}$

$$-\begin{bmatrix} \nabla\cdot(D\nabla_\perp(\circ)) - (v_r - v_{iz} - \alpha)T_g^{-1}\circ \\ \nabla\cdot\left(\eta\nabla(y_1^{-1}\circ)\right) + (v_{cx} + v_{iz} + \alpha)\rho_g\circ \end{bmatrix} \begin{bmatrix} \bar{z}_1 \\ \bar{z}_2 \end{bmatrix} = \begin{bmatrix} J^{-1}\mathcal{G}_1 \\ J^{-1}\mathcal{G}_2 \end{bmatrix}. \tag{35}$$

B. Fully Implicit Method

Now all terms will be treated implicitly. The implicit term, $\vec{\mathcal{F}}_I$, now contains the inviscid and viscous fluxes as well as the source term

$$\vec{\mathcal{F}}_I = -\mathrm{N}^{\mathrm{T}}\vec{\boldsymbol{\mathcal{E}}} + \mathrm{N}^{\mathrm{T}}\vec{\boldsymbol{\mathcal{F}}} + J\boldsymbol{\mathcal{S}}. \tag{36}$$

The nonlinear function, $\mathcal{G}$, keeps the same notation but with $\vec{\mathcal{F}}_I$ now also containing the inviscid flux.

The source Jacobian, $\mathbf{A} = \frac{\partial \boldsymbol{\mathcal{S}}}{\partial \mathbf{U}}$, can then be written with preconditioner operator actions as

$$\mathbf{A} = \begin{bmatrix} (v_r - v_{iz})\circ & 0 \\ \nabla_\parallel\left(\frac{T_g}{Jm_g}\circ\right) - V_{\parallel i}(v_{cx} + v_r)\circ & (-v_{cx} - v_{iz})\circ \end{bmatrix}. \tag{37}$$

The convective flux Jacobian, $\mathbf{B} = \frac{\partial \boldsymbol{\mathcal{E}}}{\partial \mathbf{U}}$, is given by

$$\mathbf{B} = \begin{bmatrix} 0 & J\nabla\cdot\left(\mathbf{b}J^{-1}\circ\right) \\ -J\nabla\cdot\left(\left(J^{-1}\left(\frac{y_2^2}{y_1^2}\right)\mathbf{b} - D\frac{y_2}{y_1^2}\nabla_\perp\left(\frac{y_1T_g}{J}\right)\right)\circ\right) - J\nabla\cdot\left(D\frac{y_2}{y_1}\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) & J\nabla\cdot\left(\left(\left(\frac{2y_2}{Jy_1}\right)\mathbf{b} - \frac{D}{y_1}\nabla_\perp\left(\frac{y_1T_g}{J}\right)\right)\circ\right) \end{bmatrix}, \tag{38}$$

The diffusive flux Jacobian, $\mathbf{C} = \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \mathbf{U}}$, is equal to

$$\mathbf{C} = \begin{bmatrix} J\nabla\cdot\left(D\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) - (v_r - v_{iz} - \alpha)\circ & 0 \\ J\nabla\cdot\left(\eta\nabla\left(\frac{y_2}{y_1^2}\circ\right)\right) - J\nabla\cdot\left(\left(\frac{\partial\eta}{\partial y_1}\nabla\frac{y_2}{y_1}\right)\circ\right) & J\nabla\cdot\left(\eta\nabla(J^{-1}y^{-1}\circ)\right) \end{bmatrix}. \tag{39}$$

Gathering all of the derivatives provides

$$\mathcal{P}(\mathbf{y}) = \frac{\partial \mathcal{G}}{\partial \mathbf{y}} = \begin{pmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} \\ \mathcal{P}_{21} & \mathcal{P}_{22} \end{pmatrix}, \tag{40}$$

with the matrix entries

$$\mathcal{P}_{11} = -J\nabla\cdot\left(D\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) - (v_r - v_{iz} - \alpha)\circ, \tag{41}$$

$$\mathcal{P}_{12} = J\nabla\cdot\left(\mathbf{b}J^{-1}\circ\right) \tag{42}$$

$$\mathcal{P}_{21} = -J\nabla\cdot\left(\left(J^{-1}\left(\frac{y_2^2}{y_1^2}\right)\mathbf{b} - D\frac{y_2}{y_1^2}\nabla_\perp\left(\frac{y_1T_g}{J}\right)\right)\circ\right) - J\nabla\cdot\left(D\frac{y_2}{y_1}\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) \tag{43}$$

$$+ J\nabla\cdot\left(\eta\nabla\left(\frac{y_2}{y_1^2}\circ\right)\right) - J\nabla\cdot\left(\left(\frac{\partial\eta}{\partial y_1}\nabla\frac{y_2}{y_1}\right)\circ\right) + \nabla_\parallel\left(\frac{T_g}{Jm_g}\circ\right) - V_{\parallel i}(v_{cx} + v_r)\circ, \tag{44}$$

$$\mathcal{P}_{22} = J\nabla\cdot\left(\left(\left(\frac{2y_2}{Jy_1}\right)\mathbf{b} - \frac{D}{y_1}\nabla_\perp\left(\frac{y_1T_g}{J}\right)\right)\circ\right) - J\nabla\cdot\left(\eta\nabla(J^{-1}y_1^{-1}\circ)\right) + (v_{cx} + v_{iz} + \alpha)\circ\,. \tag{45}$$

This formulation is used because the derivatives can be grouped into hyperbolic, elliptic, and source terms. Note that in this study, only the elliptic and source terms are implemented. That is, the derivatives above are reduced to

$$\mathcal{P}_{11} = -J\nabla\cdot\left(D\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) - (v_r - v_{iz} - \alpha)\circ \tag{46}$$

$$\mathcal{P}_{12} = 0 \tag{47}$$

$$\mathcal{P}_{21} = -J\nabla\cdot\left(D\frac{y_2}{y_1}\nabla_\perp\left(\frac{T_g}{J}\circ\right)\right) + J\nabla\cdot\left(\eta\nabla\left(\frac{y_2}{y_1^2}\circ\right)\right) - V_{\|i}(v_{cx} + v_r)\circ \tag{48}$$

$$\mathcal{P}_{22} = -J\nabla\cdot\left(\eta\nabla(J^{-1}y_1^{-1}\circ)\right) + (v_{cx} + v_{iz} + \alpha)\circ\ . \tag{49}$$

As with the implicit-explicit method, introduce $\boldsymbol{z} = \left(JT_g^{-1}\bar{z}_1, Jy_1\bar{z}_2\right)$ for application to the preconditioner's action. The current implementation of the elliptic solver cannot handle the different coefficients in front of the action in $\mathcal{P}_{11}$ and $\mathcal{P}_{21}$, and so the derivative is split into

$$J\nabla\cdot\left(\eta\nabla\left(\frac{y_2}{y_1^2}z_1\right)\right) = J\nabla\cdot\left(\eta\nabla\left(\frac{y_2}{y_1^2}\frac{J}{T_g}\bar{z}_1\right)\right) \tag{50}$$

$$= J\nabla\cdot\left(\eta\frac{y_2}{y_1^2}\frac{J}{T_g}\nabla\bar{z}_1\right) + J\nabla\cdot\left(\left(\eta\nabla\frac{y_2}{y_1^2}\frac{J}{T_g}\right)\bar{z}_1\right) \tag{51}$$

and dropping the hyperbolic part in Eq. (51) gives

$$\mathcal{P}_{21} = -J\nabla\cdot\left(D\frac{y_2}{y_1}\nabla_\perp(\bar{z}_1)\right) + J\nabla\cdot\left(\eta\frac{y_2}{y_1^2}\frac{J}{T_g}\nabla\bar{z}_1\right) - V_{\|i}(v_{cx} + v_r)\frac{J}{T_g}\bar{z}_1\,. \tag{52}$$

Other entries in the new preconditioner are

$$\mathcal{P}_{11} = -J\nabla\cdot(D\nabla_\perp(\bar{z}_1)) - (v_r - v_{iz} - \alpha)\frac{J}{T_g}\bar{z}_1 \tag{53}$$

$$\mathcal{P}_{12} = 0 \tag{54}$$

$$\mathcal{P}_{22} = -J\nabla\cdot(\eta\nabla(\bar{z}_2)) + (v_{cx} + v_{iz} + \alpha)J\rho_g\bar{z}_2\,. \tag{55}$$

VI. Results and Discussion

The new preconditioner is implemented in the COGENT framework. The accuracy and performance of the new preconditioner are assessed. A slab configuration representing a 2D section in the longitudinal-radial plane of a cylindrical coordinate system with the coinciding origin at the central symmetry axis of the edge geometry (or the well-known tokamak) is used. The test configurations are first described, then results are presented and discussed.

As mentioned, a 2D slab geometry, shown in Fig. (2), representative of a section in a tokamak is used. This consists of a rectangular geometry with a width of 1.1 m and a height of 2 m. The density is initialized to

$$\rho_0 = 1.0 + \frac{(y-2)^2}{4}\,, \tag{56}$$

the parallel neutral velocity is initialized to zero, and the parallel ion velocity is initialized to

$$V_{\|g,0} = \sin(y\pi)\,. \tag{57}$$

After solution time $t = 0.125$, the longitudinal profile of neutral density and neutral parallel velocity develops as shown in Fig. (3). As seen from the figure, the solution profiles obtained by the new preconditioner reproduce those by the old preconditioner, which is expected for a properly working preconditioner. To further verify the solution accuracy, error norms are measured. The grid convergence rate should match the existing preconditioner, which is a rigorous test to ensure that the new preconditioner is properly implemented and applied. The L_p-norm ($p = 1, 2, \infty$) is listed in Tbl. (1).

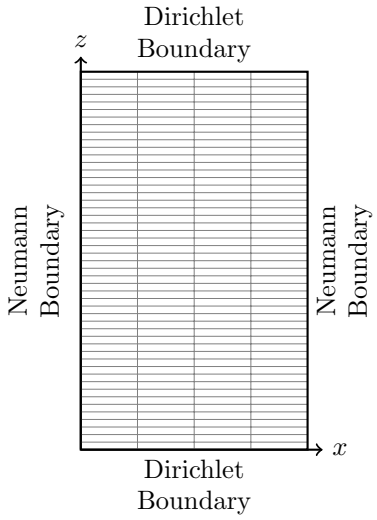

Fig. 2 Representation of the slab geometry with boundary conditions. The mesh is stretched in physical space.

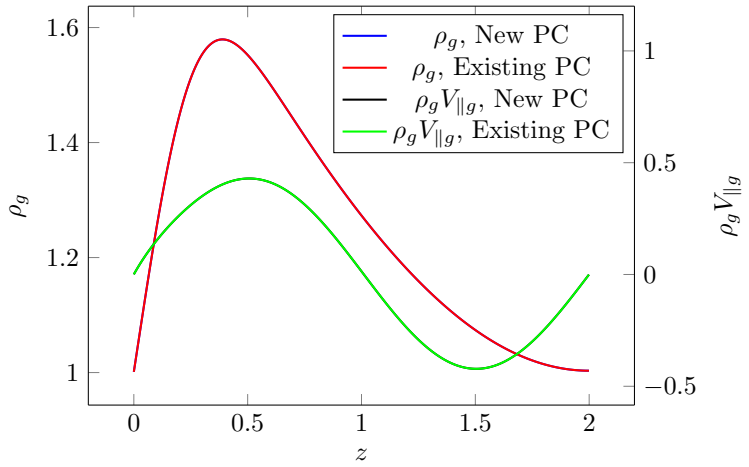

Fig. 3 Profiles of the density and the parallel velocity for the neutrals along the z-direction (longitudinal) at the center of the radial coordinate. The two preconditioners are identically similar.

Table 1 A Richardson extrapolation verifies the order of accuracy of the new preconditioner. As the mesh is refined, the error trends towards second-order convergence. The errors and convergence rates for the existing preconditioner are provided for reference. Both preconditioners converge at the same rate.

New Preconditioner								
Var	L_p-norm	4×128	Rate	8×256	Rate	16×512	Rate	32×1024
ρ	L_∞	2.785e-04	2.058	6.690e-05	2.088	1.574e-05	2.113	3.640e-06
	L_1	4.954e-05	2.108	1.149e-05	2.052	2.771e-06	1.917	7.338e-07
	L_2	9.289e-05	2.095	2.174e-05	2.108	5.043e-06	1.975	1.283e-06
$\rho V_{\parallel g}$	L_∞	4.370e-03	1.008	2.173e-03	1.116	1.003e-03	1.245	4.232e-04
	L_1	2.569e-04	1.253	1.078e-04	1.240	4.562e-05	1.307	1.844e-05
	L_2	6.958e-04	1.070	3.313e-04	1.145	1.498e-04	1.260	6.255e-05
Existing Preconditioner								
Var	L_p-norm	4×128	Rate	8×256	Rate	16×512	Rate	32×1024
ρ	L_∞	2.785e-04	2.058	6.690e-05	2.088	1.574e-05	2.113	3.640e-06
	L_1	4.954e-05	2.108	1.149e-05	2.052	2.771e-06	1.917	7.338e-07
	L_2	9.289e-05	2.095	2.174e-05	2.108	5.043e-06	1.975	1.283e-06
$\rho V_{\parallel g}$	L_∞	4.370e-03	1.008	2.173e-03	1.116	1.003e-03	1.245	4.232e-04
	L_1	2.569e-04	1.253	1.078e-04	1.240	4.562e-05	1.307	1.844e-05
	L_2	6.958e-04	1.070	3.313e-04	1.145	1.498e-04	1.260	6.255e-05

A Richardson extrapolation is used to compute the errors for measuring the order of accuracy of the new preconditioner. A sequence of mesh sizes, refined from 4×128 cells up to 64×2048 cells with a refinement ratio of 2 between two consecutive meshes, are used. Clearly shown, the error reduction rate converges to second order as the mesh is refined.

For the same solution accuracy, the performance of the preconditioners is of interest. To assess the performance, two cases are run to the same solution time, one with the new preconditioner and the other with the existing preconditioner. The mesh of 4×128 cells is used and 400 time steps are taken. An absolute convergence tolerance of 1×10^{-10} and relative tolerance of 1×10^{-6} are used. Measures, such as the function evaluation, nonlinear iterations, linear iterations,

Table 2 Performance comparison between the new preconditioner and the existing preconditioner. The number of function evaluations, nonlinear iterations, and linear iterations were the same for all runs of each preconditioner, while the average wall-clock time, median wall-clock time, and one standard deviation are are measured from all 10 runs.

Measure	New Preconditioner	Existing Preconditioner	% Reduction
Function Evaluations	7008	7937	11.7
Nonlinear iterations	834	835	0.1
Linear iterations	2973	3901	23.8
Average wall-clock time (s)	32.14	31.74	-1.3
Median wall-clock time (s)	32.02	31.70	-1.0
One standard deviation (s)	0.44	0.31	-41.9

and wall-clock time, are presented in Tbl. (2). The new preconditioner requires fewer function evaluations, nonlinear iterations, and linear iterations than the existing preconditioner. The new preconditioner decreases the residual by approximately 1.7 magnitudes per iteration while the existing preconditioner decreases the residual by approximately 1.3 magnitudes per iteration. Unfortunately, the new preconditioner is currently, on average, slightly slower than the existing preconditioner. This is due to the fact that the new preconditioner is solving a block 2x2 matrix for each cell, which is more expensive than the two 1x1 equation solves at each cell performed by the existing preconditioner.

VII. Conclusion and Future Work

An analytic preconditioner for the two-field neutral model is derived and the elliptic and source terms of the preconditioner are implemented in COGENT. The new preconditioner preserves the second-order solution accuracy. This new preconditioner reduces the number of function calls, nonlinear iterations, and linear iterations compared to the existing preconditioner, but a speedup is not yet realized. Nevertheless, as a first step, the present study derives the full preconditioner and lays out the code structure for future study.

Future work will be focused on performance optimization. Moreover, the preconditioner will be applied to large-scale problems which may demonstrate more computational gain than the existing preconditioner. The inclusion of the hyperbolic term in the analytic preconditioner will be considered and its impact on performance will be investigated.

VIII. Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. DE-AC52–07NA27344.

References

[1] https://www.differ.nl/research/plasma-material-interactions, 2021.

[2] https://github.com/LLNL/COGENT, 2021.

[3] Dorf, M., *Two-Field Neutral Model*, Lawrence Livermore National Laboratory, 2021.

[4] Ghosh, D., *Implicit-Explicit Time Integration in COGENT*, Lawrence Livermore National Laboratory, April 2018.