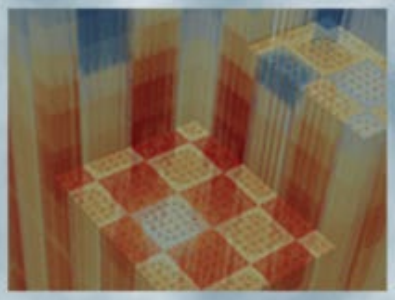
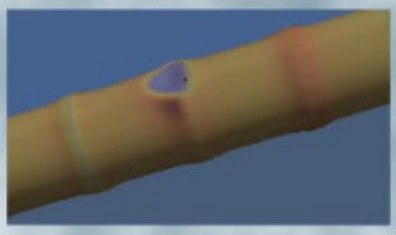
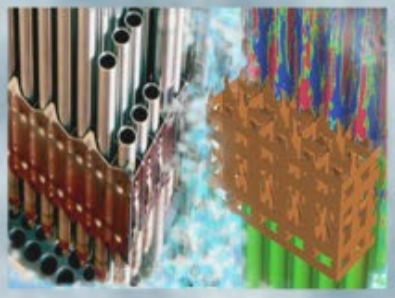
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, 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 owned rights. Reference herein to any specific commercial product, 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 any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

CASL-U-2017-1354-000







Mongoose Methods and Theory

Benjamin Collins
Oak Ridge National Laboratory

Jack Galloway
Los Alamos National Laboratory

May 15, 2017

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0	5/15/2017	All	Initial Release

Document pages that are:

Export Controlled None

IP/Proprietary/NDA Controlled None

Sensitive Controlled None

Unlimited ALL

Requested Distribution:

To: Jess Gehin, Director
Dave Kropaczek, Chief Scientist

Copy: Kevin Clarno, PHI Lead
Tom Downar, PHI Deputy
Brian Wirth, FMC Lead
David Andersson, FMC Deputy

CONTENTS

FIGURES

ACRONYMS

AO	Axial Offset
API	Application Programming Interface
CASL	Consortium for Advanced Simulation of Light Water Reactors
CIPS	CRUD-Induced Power Shift
CTF	COBRA-TF subchannel thermal-hydraulics code
COBRA-TF	Coolant Boling in Rod Arrays-Two Fluids
DOE	U.S. Department of Energy
EFPD	Effective Full Power Days
FMC	Fuel Materials and Chemistry Focus Area
LWR	Light Water Reactor
PWR	Pressurized Water Reactor
VERA	Virtual Environment for Reactor Applications

ACKNOWLEDGEMENTS

The authors of this report acknowledge the many contributions that have been made by others to reach this milestone. The results presented in this document are the culmination of years of research and development, software testing and debugging, data collection, project planning, and many other tasks performed by many CASL participants.

1. INTRODUCTION

The Consortium for Advanced Simulation of Light Water Reactors (CASL) has developed a suite of high-fidelity software and methods that is capable of simulating the full operating history of a commercial pressurized water reactor (PWR), including the loading, depletion, shuffling, and discharge of all nuclear fuel assemblies. These tools, collectively known as the Virtual Environment for Reactor Applications, or VERA, include advanced solvers and multi-physics coupling algorithms that provide the most rigorous solutions available today for steady-state nuclear reactor analysis. To demonstrate this capability, VERA has recently been used to simulate over 18 years of operation of Watts Bar Nuclear Unit 1 (WBN1), and the results of these calculations are used to quantify its performance in an industry-grade benchmarking activity.

The mission of CASL is to improve the efficiency and lifetime of the U.S. nuclear fleet by the development and deployment of advanced physics methods and software capabilities. These goals will be achieved through the application of VERA to the CASL "challenge problems", which target operational (CIPS, CILC, PCI, GTRF) and safety (DNB, RIA, LOCA) issues that have not been fully resolved by currently available tools. At the heart of these simulations is the VERA core simulator, known as VERA-CS, which provides the capabilities to simulate an operating reactor over long time scales and provides the starting and bounding conditions to the challenge problem analyses. VERA-CS can perform the same functions as currently licensed reactor core analysis tools, but at higher resolutions and with fewer approximations.

This report documents the methods and theory embedded in a new coolant-water surface chemistry code called Mongoose, which is being used by CASL in support of the CIPS challenge problem. The general-purpose deposition, growth, and densification surface chemistry capability within Mongoose is being used to model the evolution of a nickel-ferrite crud layer, support modeling the precipitation of lithium tetraborate, the major driver of CIPS in PWRs, and source of nickel, iron, and lithium metals from the corrosion of piping and heat exchangers. When the capability in Mongoose is tightly coupled to the capability in VERA-CS, whole core modelling of the reactor including the source-terms and effects of crud and lithium tetraborate deposition can be captured. This will allow CASL to model CIPS at a fundamental level in the core and begin to benchmark the capability against existing reactors with and without CIPS.

2. MONGOOSE METHODS

The development of Mongoose stemmed from the need for CASL to have a flexible, fast, and accurate prediction of crud chemistry that could easily be integrated into the existing VERA software. Previous developments for coolant chemistry were performed in MAMBA3D and MAMBA1D [1]. While many of the physical models developed in Mongoose have their origins in the MAMBA3D code, the underlying infrastructure and many of the numerical methods have been updated to reduce memory burden, improve computational efficiency, and enable rapid prototyping of new physics. In addition, the source-term models for crud are due to the erosion of metals from the piping and heat exchangers. Unlike MAMBA, Mongoose was designed to solve a general-purpose water-metal surface thermo-chemistry problem, of which crud is one application. Because this initial report is focused on CASL need for a replacement to MAMBA, the following sections focus on this nuclear-specific aspect of Mongoose.

2.1 Mongoose Architecture and Design

The Mongoose infrastructure contains three basic components. The first component defines a full height cylinder, which would represent a fuel pin for a crud application. This cylinder will have a fixed azimuthal and axial grid provided by client code. Each of these azimuthal and axial grid points will contain the second component which defines the radial mesh. Since the crud, or other surface chemistry precipitate, grows radially over time, the number of mesh elements at any point in time is unknown and the radial direction must be treated differently than the azimuthal and axial directions. When a radial mesh zone becomes active, the third component which defines the physics of a node, is created. The crud node contains all the dominant physical phenomenon that occur. Figure 1 shows the interaction of the three different levels.

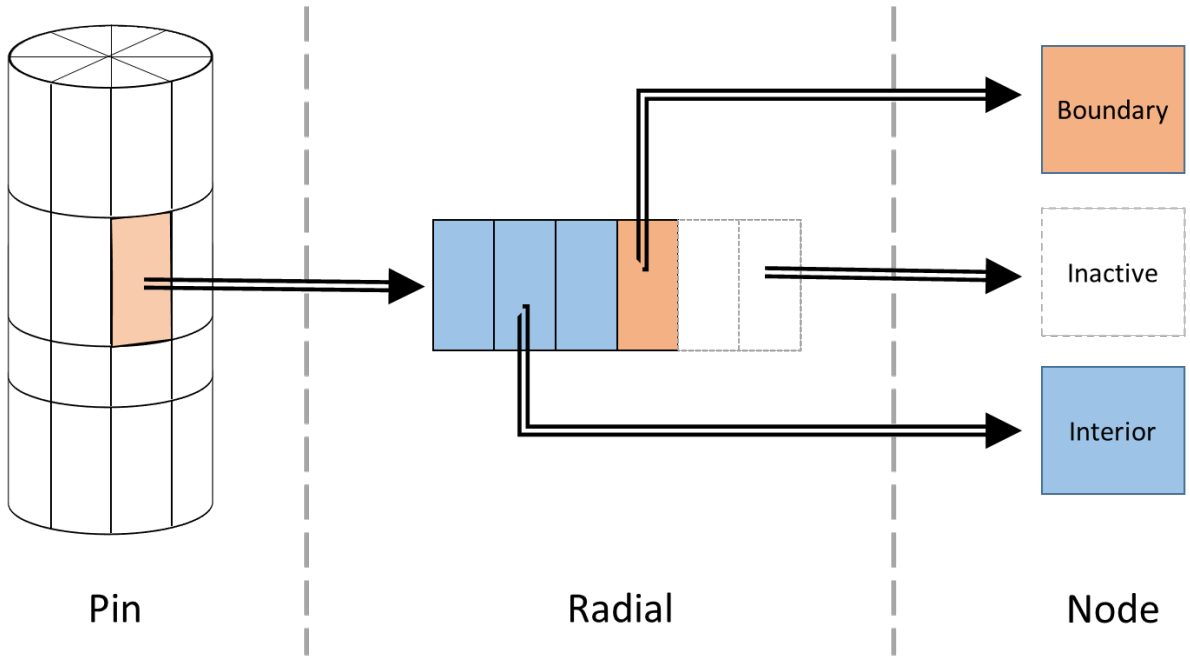


Figure 1: Mongoose Component Hierarchy

2.1.1 Pin Component

Each component in this hierarchy has its own role. The pin component oversees defining the azimuthal and axial mesh and owns the 2D grid of radial components. Currently the azimuthal

grid assumes uniformly spaced discretization but the axial mesh allows for non-uniform axial heights.

Additionally, this component manages all the input and output data for the pin. The client code that couples with each Mongoose pin will define the coolant chemistry conditions for each pin, which include concentrations of the chemical constituents in the coolant. The client code must also define:

- the radial heat flux at the surface of the cladding, the surface temperature on the outside of the crud layer
- turbulent kinetic energy on the surface of the crud layer, and
- the bulk coolant temperature for every azimuthal and axial mesh component.

The pin component controls passing data to the respective radial components. For output, the pin component provides accessors to the client code to compute crud thickness, crud surface density, boron surface density, the thermal resistance of the crud, and the mass evaporation rate for every azimuthal and axial mesh element. The pin component will soon provide an interface which can provide 3D distributions of the data for custom post-processing including data visualization to the client code.

Another key function of the pin component is to control the iteration of Mongoose. There are two key features that are provided: 1) time step control of the solution and 2) the ability to restart the solution at a previous time step. The time step control of the solution involves taking the time steps provided by the client code and decomposing them into substeps which are defined by the user. In each substep, all radial component are advanced for the substep size. Then the temperature in the entire model is updated. Once this is complete, the next substep is performed until the requested time step is complete.

Another feature that is available is the ability to rerun the same time step multiple times with different boundary conditions. This is done by storing the state of all the radial components at the beginning of the desired time step in a restart array. This information can then be used to rerun a time step. The default behavior of Mongoose currently is that the time step is resolved each time the client code requests a solve until the client code calls the *advance_state* method on the pin component. When this method is called, the restart array is updated with the state from the last solution. This is done because previous coupling activities [2] showed that it is necessary to iterate a current time step between neutronics, thermal hydraulics, and crud chemistry to converge the overall solution. It may be desirable in the future to update the behavior to have the client code decide when to rewind the solution. The expected modifications required to do this are very straightforward and could be done very quickly if desired.

The only physics component that is directly implemented on the pin component is the solution of the heat conduction to obtain temperature. The physics of the temperature solve are discussed in much greater detail in section 2.6 which details the heat transport methodology. The current implementation only contains a radial heat conduction model, which is implemented in the radial component (and direction), but it is anticipated that a 3D heat transfer solution might be required for cases with more extreme crud growth.

2.1.2 Radial Component

The radial component was developed to handle the dynamic growth of the crud layer during reactor operation. The radial component is primarily intended to be called from the pin component but it is feasible that it could also be used in place of MAMBA1D to provide subgrid on an arbitrary mesh. Minor refactors on the interface would be needed for this to occur, but it is

may prove necessary when coupling directly to a code with an unstructured grid (e.g. StarCCM+).

The implementation of the radial component is done by creating a 1-D grid of pointers to node components. The size of the grid is defined by the client code but it is conceivable that the size could be updated dynamically. Since the grid only stores a list of pointers, very minimal memory is used even though a fixed sized array is being used. The client code also provides a fixed radial mesh spacing. As the radial mesh grows, new node elements are allocated and activated in the solution sequence.

In addition to mesh control, the radial component controls all the physics solutions in the radial direction. When the pin component provides a substep to solve, the radial component will:

- Compute the equilibrium chemistry on the surface of the crud (Section 2.3)
- Compute local vapor flow rates (Section 2.5)
- Compute local diffusion lengths based on radial conditions (Section 2.5)
- Solve kinetics on each interior node (Section 2.5)
- Solve kinetics on the surface node (Section 2.4)
- Test for fully filled surface node and extend mesh if needed

Several of these steps involve integrating quantitates in the radial direction which require looking at local temperatures, mass evaporation rates, and current state of the internal crud chemistry to determine quantities.

The radial component contains the main functions which will compute the integral quantities which are needed for output of the pin component such as crud thickness, surface densities, and mass evaporation rates. Additionally, the radial conduction solve option, mentioned in the pin component section, is implemented in the radial component. The radial component also implements a *copyFrom* method which is used in the rewind feature to move data between the current state and the restart state of the code.

2.1.3 Node Component

As mentioned above, the node component is where most of the physical models are implemented since much of the dominant physics does not require neighboring information. As seen in Figure 1, there are three states the node component can exist in; inactive, boundary, and interior. Inactive nodes have no allocated memory but are useful to conceptualize to understand the behavior and limitations of the code.

The boundary and interior node types have identical data associated. The major difference is the kinetics solve relies on fundamentally different physics which are described in Sections 2.4 and 2.5 respectively. The boundary node type also has a method to compute surface coolant chemistry which is discussed in Section 2.3. A logical variable is stored to distinguish between the two node types and a method exists to convert between the two.

Additionally, the node component has methods to compute local boiling heat transfer coefficients, local boiling heat flux, local thermal conductivity, and local thermal diffusivity. A *copyFrom* method also exists to support the rewind feature on the pin and radial components.

2.2 Mongoose Physical Properties

The first step in the development of Mongoose was to create a common library for chemical and physical properties of crud and the chemical constituents in the coolant. A library called *crud_utils* was created for Mongoose, and MAMBA3D was modified to use it. As part of the development of this library, the software interfaces are now well defined and documented in the code and each routine has been placed under a thorough unit testing framework available in Futility [3].

The first module in *crud_utils* contains several functions for water properties. These include thermo-physical properties, activity coefficients, and equilibrium constants. There are also several functions that provide analytic derivatives of other quantities and are used in the Jacobian matrices for nonlinear solves. Future work here should consider ensuring the thermo-physical properties agree well with CTF and Star-CCM.

The second module in *crud_utils* contains properties specific to crud formation. These include constants for molar masses of several key isotopes, elements, and compounds including Ni, Fe, Li, B, B-10, B-11, O, Nickle Ferrite, and Lithium tetraborate. Additionally, several variables which were previously input into MAMBA3D have been added with predefined values including:

- prefactors and activation energies for computing Arrhenius rates coefficient inside and outside of the crud
- boiling deposition rate constant for nickel-ferrite
- base diffusion coefficients of major chemical species in coolant
- initial porosity of the crud
- solid density of nickel-ferrite
- chimney surface density
- representative chimney radius
- chimney heat transfer coefficient.

These coefficients have representative defaults but can be modified by the user for sensitivity analysis or further calibration of the models. In addition to these constants, functions are available to calculate the equilibrium constant for multiple reactions, activity coefficients, and thermo-physical properties of nickel-ferrite.

The *crud_utils* library also provides initial functionality for computing the source term for nickel-ferrite in the primary system which is currently being implemented into the system mass balance that is available in CTF.

2.3 Coolant Thermodyamics

The thermodynamic models from MAMBA3D have also been reproduced in *crud_utils* to share common functionality. Several soluble and solid phase species are currently included in this model which includes:

- Borate species – $B(OH)_3, B(OH)_4^-, B_2O(OH)_5^-, B_3O_3(OH)_4^-$
- Lithium species – $LiOH, Li^+$
- Water and hydrogen dissolution species – H_2O, OH^-, H^+, H_2
- Iron ionic species – $Fe^{2+}, Fe(OH)_2$
- Nickel ionic species – $Ni^{2+}, Ni(OH)_2$
- Solid species – $NiFe_2O_4, Li_2B_4O_7$

The computation of these various species occurs in three main parts. The first stage computes the relationship for boric acid and borate species, the lithium species, and water dissolution. Because the boric acid concentration is significantly higher than nickel and iron (ppm vs ppb), these two systems are decoupled and the hydrogen and hydroxide species are computed based on the boric acid speciation. A nonlinear solve is performed to ensure boron mass balance and charge balance in order to obtain the concentrations of all boron, lithium, and hydrogen soluble species.

Once the boron species are obtained, the hydroxide and hydrogen concentrations are treated as known quantitates and the iron and nickel speciation can be directly obtained. Additionally, the nickel ferrite precipitation constant can be obtained. Finally, the lithium tetraborate precipitation number is obtained based on the concentrations previously calculated.

2.4 Crud Surface Kinetics

The growth of the crud layer on the surface of the cladding is driven by the surface deposition of nickel ferrite particulate. The kinetic equation that drives the surface deposition is

$$\frac{dC_{NiFe_2O_4}}{dt} = \left(k^p_{s,non-boil} + k^p_{s,boil} q''_{s,boil}\right) C^p_{NiFe_2O_4,cool} - \gamma_{s,e} k_{TKE}. \quad [1]$$

Here the nickel ferrite concentration in the crud layer is growing from both non-boiling deposition and boiling enhanced deposition. The deposition rate coefficient for non-boiling is described as an Arrhenius rate where the prefactor and activation energy are constants in *crud_utils* which can be modified. The deposition rate caused by boiling has two terms; the deposition coefficient $k^p_{s,boil}$ which has units of $\frac{cm^3}{J}$ and the boiling heat flux on the surface of the crud layer $q''_{s,boil}$ which has units of $\frac{W}{cm^2}$. Both of these deposition rates are multiplied by the nickel ferrite particulate concentration in the coolant. The last term represents the suppression of growth caused by erosion on the surface of the crud layer. The model used to account for erosion is a scaling, $\gamma_{s,e}$, applied to the predicted turbulent kinetic energy, k_{TKE}. Since all of these terms are independent of the actual concentration of nickel ferrite in the crud layer, the analytic solution is easily obtained

$$C_{NiFe_2O_4}(t+\delta t) = C_{NiFe_2O_4}(t) + \delta t \left(\left(k^p_{s,non-boil} + k^p_{s,boil} q''_{s,boil}\right) C^p_{NiFe_2O_4,cool} - \gamma_{s,e} k_{TKE} \right). \quad [2]$$

Mongoose assumes that particulate nickel ferrite is the only compound that contributes to the growth of the crud layer surface. The crud structure is assumed to grow to a set porosity which defaults to 70%. Once the concentration of nickel ferrite in the crud layer becomes sufficient to fill the entire node at 70%,

$$C_{NiFe_2O_4} \geq \frac{\eta \rho_{NiFe_2O_4}}{M_{NiFe_2O_4}}, \quad [3]$$

the node is switched from a boundary node to an internal node. Any remaining concentration is moved into the newly activated boundary node and the local surface concentration of soluble nickel, iron, boron, lithium, and hydrogen gas are the initial concentrations which get fed into the internal kinetics routines.

It should be noted that the extension of the surface kinetics to account for deposition of nickel metal or nickel oxide would result in a set of 3 coupled differential equations which may not have an analytic solution. In this case, the ODE solver interface discussed in the next section would be required.

2.5 Crud Internal Kinetics

While the surface kinetics only considers the growth of nickel ferrite, the internal kinetics allows several species to change including nickel ferrite ($C_{NiFe_2O_4}$), soluble nickel (C_{Ni}), soluble iron (C_{Fe}), soluble boron (C_B), soluble lithium (C_{Li}), soluble hydrogen (C_{H_2}), and lithium tetraborate ($C_{Li_2B_4O_7}$). The precipitation thresholds of nickel ferrite and lithium tetraborate are computed using the thermodynamic model discussed previously. The internal deposition of nickel ferrite is governed by

$$\frac{\boldsymbol{dC_{NiFe_2O_4}}}{\boldsymbol{dt}} = \begin{cases} \boldsymbol{k_i\eta\left(C_{Ni_s}C_{Fe_s}^2 - p_{NiFe_2O_4}C_{cool}^3\right)} & \boldsymbol{C_{Ni_s}C_{Fe_s}^2 > p_{NiFe_2O_4}C_{cool}^3} \\ \boldsymbol{0} & \end{cases}, \quad [4]$$

where k_i is the Arrhenius rate of deposition based on the predefined prefactor and activation energy for internal kinetics, η is the porosity, $p_{NiFe_2O_4}$ is the precipitation parameter for nickel ferrite and C_{cool} is the concentration of bulk coolant. The conditional check ensures that nickel ferrite is only deposited and not removed.

The second equation modeled accounts for the porosity (η) of the node. As nickel ferrite deposits, the porosity is decreased

$$\frac{\boldsymbol{d\eta}}{\boldsymbol{dt}} = -\frac{\boldsymbol{M_{NiFe_2O_4}}}{\boldsymbol{\rho_{NiFe_2O_4}}}\frac{\boldsymbol{dC_{NiFe_2O_4}}}{\boldsymbol{dt}}. \quad [5]$$

If nickel metal or nickel oxide compounds are added, this constraint will need to be increased to account for these terms. The lithium tetraborate precipitation also affects the porosity but is treated specially and will be addressed later in this section.

There are five soluble components, soluble nickel (C_{Ni}), soluble iron (C_{Fe}), soluble boron (C_B), soluble lithium (C_{Li}), and soluble hydrogen (C_{H_2}), which all have the same transport equation

$$\frac{\boldsymbol{dC_x}}{\boldsymbol{dt}} = \frac{\boldsymbol{1}}{\boldsymbol{\delta r}}\left(\boldsymbol{F_{boil,x} - \eta F_{diff,x} - F_{vap,x}}\right). \quad [6]$$

The species concentration change is dependent on the inflow of fresh coolant caused by local subcooled boiling conditions, diffusion of the species back out to the coolant, and carryover of the species in the vapor. The inflow of fresh coolant is balanced by the outflow of coolant due to localized boiling. The local flux is the local boiling velocity multiplied by the coolant concentration of the soluble species

$$\boldsymbol{F_{boil,x}} = \frac{\boldsymbol{q_{boil}'''\delta r}}{\boldsymbol{h_{fg}\rho_f}}\boldsymbol{C_{x,cool}}, \quad [7]$$

where q_{boil}''' is the boiling heat flux, h_{fg} is the latent heat of vaporization, and ρ_f is the density of the fluid. The diffusive flux accounts for molecular diffusion between the node and the bulk coolant. It is modeled using a standard diffusion relationship,

$$\boldsymbol{F_{diff,x}} = \boldsymbol{D_x}\frac{\boldsymbol{C_{x,cool} - C_x}}{\boldsymbol{\Delta L}}, \quad [8]$$

where D_x is the diffusion coefficient and ΔL is the distance to the coolant. The vapor flux occurs when the species is entrained in the exiting vapor. This is only modeled for soluble boron and soluble hydrogen:

$$\boldsymbol{F_{vap,x} = v_s C_x}, \quad [9]$$

where v_s is the local velocity of the steam itself.

Special treatment for the soluble components exists if there is not boiling in the local node. If there is no local boiling, but a flux of the species to nodes below the current node larger than the diffusive component, it is assumed that there is no change in concentration of that species. The reason for this is unknown but consistent with the treatment in MAMBA. Otherwise, the diffusive term is considered but instead of always relating to the coolant concentration, the code allows for the diffusion to be between the local node and a trapped node caused by localized precipitation or dryout. If a trapped node is not detected, diffusion occurs with the coolant.

Equations [4], [5], and [6] represent 7 coupled ordinary differential equations (ODE) where the unknowns are highlighted in red to help identify the coupling. To solve this coupled system, an ODE solver focused the implementation of the backward difference formula (BDF) has been implemented into Futility. This method implements up to 5th order BDF methods and is solved using a constant time step. BDF methods belong to the implicit linear multistep method family because it uses previously computed time steps to predict the next time step. We consider a generic set of ODEs

$$\frac{d\vec{y}}{dt} = \boldsymbol{F}(t, \vec{y}), \quad [10]$$

where $\vec{y}$ is the solution vector and F is the nonlinear response that relates the solution vector and current time to its derivative at that time. In Futility, this nonlinear response is provided by the client code which has inputs of t and $\vec{y}$, and will return the derivative of $\vec{y}$. The general form of the BDF formula can be written as

$$\boldsymbol{y_{n+1} - h\beta F(t_{n+1}, y_{n+1}) = \sum_{k=0}^{N_{BDF}} \alpha_k y_{n-k}}, \quad [11]$$

where n is the current iteration, h is the time step size, and α and β are constants of the BDF method depending on the BDF order requested. BDF methods are implicit and require the solution of a nonlinear set of equations to obtain the value of y at the next time step. A modified Newton method is used to solve these equations. First the BDF formula is rewritten

$$(\mathbb{I} - \boldsymbol{h\beta}\mathbb{F})\vec{y} = \sum_{k=0}^{N_{BDF}} \boldsymbol{\alpha_k y_{n-k}} = \vec{b}, \quad [12]$$

where $\mathbb{I}$ is the identity matrix, $\mathbb{F}$ is the nonlinear response matrix, and $\vec{b}$ is a collection of the entire RHS of the BDF formula. Also, the subscript was dropped from $\vec{y}$ to increase clarity. Newtons method can be applied to this nonlinear system by rewriting it as

$$\left(\mathbb{I} - \boldsymbol{h\beta}\frac{d\mathbb{F}}{d\boldsymbol{y}}\right)\overrightarrow{\Delta \boldsymbol{y}} = \vec{b} - (\mathbb{I} - \boldsymbol{h\beta}\mathbb{F})\vec{y},$$
$$(\mathbb{I} - \boldsymbol{h\beta}\mathbb{J})\overrightarrow{\Delta \boldsymbol{y}} = -\vec{r}, \quad [13]$$

where $\mathbb{J}$ is the Jacobian matrix which is computed numerically by finite difference and $\vec{r}$ is the residual vector. Since this is a small linear system, the matrix inverted using an LU factorization

which means that only the residual must be evaluated every iteration. The Jacobian is estimated using a finite difference scheme and is only evaluated every 20 time steps.

While this methodology provides a robust solution, a very small time step is required to resolve all of the time scales of the kinetics inside the crud. Because of this, the majority of the time spent inside of Mongoose is solving this system of ODEs. It is believed that a variable time step implementation of BDF will provide a considerably more efficient implementation. For this reason, the Sundials cvode ODE solver [4] has been implemented as a secondary option in Futility. While the ODE solve still accounts for a large majority of the solution time, the use of Sundials significant improves the overall performance of the code.

As mentioned previously, the precipitation of lithium tetraborate is not directly modeled as part of the system of ODEs above. The reason for this is the time scale for precipitation is very fast and can be treated as an instantaneous process. Once the ODE solve is complete, the concentrations of lithium and boron are sufficient to precipitate

$$C_B^4 C_{Li}^2 > p_{Li_2B_4O_7} C_{cool}^6, \quad [14]$$

where $p_{Li_2B_4O_7}$ is the lithium tetraborate precipitation factor. If this condition is true, 99% of the remaining porosity is filled with lithium tetraborate at a density of 2.487 g/cc. Once this occurs, internal kinetics is disabled for the node.

Lastly, a dryout model also exists which will test for lithium tetraborate precipitation. This model is activated if the mass flow rate of vapor out of a node is greater than the mass flow into the node. If this occurs, the thermodynamics are tested at a void fraction of 1% and 99% to see if either condition results in the condition in which precipitation will happen as shown in [14]. If precipitation occurs, 50% of the remaining porosity is filled with lithium tetraborate and the node is no longer active.

2.6 Crud Heat Transport

A key component to the evolution of the crud growth is knowing the temperature distribution throughout the crud layer. The steady state heat conduction equation in 3D is

$$\nabla \cdot k \nabla \mathbf{T} = \mathbf{q}_{\mathbf{sink}}''', \quad [15]$$

where k is the thermal conductivity which is a function of local temperature and porosity and q_{sink}''' is the local heat sink caused by boiling. The heat sink can be described as

$$\mathbf{q}_{\mathbf{sink}}''' = \begin{cases} 2\pi r_{chim} \mu(\eta)\, h_{chim} \rho_{chim} (T - T_{sat}) & T > T_{sat} \\ 0 \end{cases}, \quad [16]$$

where r_{chim} is the characteristic radius of a chimney, h_{chim} is the boiling heat transfer coefficient of a chimney, ρ_{chim} is the surface density of chimneys, and $\mu(\eta)$ is the permeability of the crud which is a function of the porosity. For simplicity, it can be rewritten as

$$\mathbf{q}_{\mathbf{sink}}''' = h_{snb}(T - T_{sat}). \quad [17]$$

The heat conduction equation becomes

$$\nabla \cdot k \nabla \mathbf{T} - \mathbf{h}_{\mathbf{snb}} \mathbf{T} = -h_{snb} T_{sat}. \quad [18]$$

The heat conduction equation is discretized using a finite volume scheme. The first step is to integrate the equation over the node volume

$$\int \nabla \cdot \boldsymbol{k}\nabla \mathbf{T}\mathbf{dV} - \int \mathbf{h_{snb}}\mathbf{T}\mathbf{dV} = -\int \boldsymbol{h_{snb}}\boldsymbol{T_{sat}}\boldsymbol{dV}. \quad [19]$$

The first term can be rewritten as a surface integral and the second two terms can be approximated as volume averages

$$\sum_{\mathbf{surf}} \boldsymbol{k}\nabla \mathbf{T} - \mathbf{h_{snb}}\mathbf{T}\,\mathbf{V} = -\boldsymbol{h_{snb}}\boldsymbol{T_{sat}}\boldsymbol{V},$$
$$\sum_{\mathbf{surf}} \boldsymbol{q_{surf}} + \mathbf{h_{snb}}\mathbf{T}\,\mathbf{V} = \boldsymbol{h_{snb}}\boldsymbol{T_{sat}}\boldsymbol{V}. \quad [20]$$

The boundary conditions for the crud layer are both defined by the client code. The heat flux is prescribed on the inside surface. The surface temperature is also provided on the outside of the crud layer.

As mentioned earlier, the original MAMBA3D code solved the heat conduction in 3-D. To increase performance and decrease speed of development, a 1-D radial heat conduction is currently implemented into Mongoose. A 3-D conduction solve will be added in the second phase of development to capture azimuthal and axial transport effects. This simplifies the heat conduction equation to

$$\boldsymbol{V} = \mathbf{r}\,\boldsymbol{\delta\theta}\,\boldsymbol{\delta z}\,\boldsymbol{\delta r},$$
$$A_{out} = \delta\theta\ \delta z\left(r + \frac{\delta r}{2}\right)$$
$$A_{in} = \delta\theta\ \delta z\left(r - \frac{\delta r}{2}\right) \quad [21]$$
$$\boldsymbol{q''_{in}}\boldsymbol{A_{in}} - \mathbf{q''_{out}}\mathbf{A_{out}} - \mathbf{h_{snb}}\mathbf{T}\,\mathbf{V} = -\boldsymbol{h_{snb}}\boldsymbol{T_{sat}}\boldsymbol{V},$$

where A_{in} is the surface area of the inner radial surface, A_{out} is the surface area of the outer radial surface, and q'' are the corresponding heat fluxes on that surface. To obtain the heat flux on the inner and outer surface of the node, the derivative is approximated using a central difference approximation using the harmonic mean average of the thermal conductivity

$$\mathbf{q''_s} = \frac{\mathbf{2}\boldsymbol{k^+k^-}}{\boldsymbol{\delta r}(\boldsymbol{k^+} + \boldsymbol{k^-})}(\boldsymbol{T^-} - \boldsymbol{T^+}), \quad [22]$$

where k^+, k^- represent the thermal conductivity on the positive and negative side of the surface and likewise T^+, T^- are the node average temperatures on each side of the interface. For simplicity, the surface harmonically averaged thermal conductivity will be written as

$$\widetilde{\boldsymbol{k_s}} = \frac{\mathbf{2}\boldsymbol{k^+k^-}}{\boldsymbol{\delta r}(\boldsymbol{k^+} + \boldsymbol{k^-})}. \quad [23]$$

The leads to the following radial heat conduction equation

$$\widetilde{\boldsymbol{k_{in}}}\boldsymbol{A_{in}}(\boldsymbol{T_{i-1}} - \boldsymbol{T_i}) - \widetilde{\boldsymbol{k_{out}}}\mathbf{A_{out}}(\boldsymbol{T_i} - \boldsymbol{T_{i+1}}) - \mathbf{h_{snb}}\mathbf{T_i}\,\mathbf{V} = -\boldsymbol{h_{snb}}\boldsymbol{T_{sat}}\boldsymbol{V}, \quad [24]$$

where T_i represents the temperature for the current node, T_{i-1}, T_{i+1} represent the neighboring node below and above the current node respectively. This relationship forms a tridiagonal linear system of equations that handles the conduction and boiling term.

This linear system must be solved multiple times because the thermal conductivity and the subcooled boiling heat transfer coefficient are both temperature dependent. In practice, the solution is converged after a few nonlinear updates of the coefficients. The linear solve here is solved using a LU factorization available for the tridiagonal solver in Futility. While the actual solve is very fast and efficient, the nature of the 1-D conduction solve is that several linear systems are solved for each azimuthal and axial mesh every time step, each with a different number of radial mesh elements. For this reason, the size of the linear system cannot be reused. A significant overhead was discovered when creating and destroying linear system objects for each azimuthal and axial mesh element each time step. For this reason, a current work around is to solve a linear system the size of the maximum number of radial elements. The tridiagonal matrix is used in the upper left corner of the matrix and the identity matrix is entered for the remainder of the matrix. While this prevents the need to create and destroy linear solver objects, the computational burden to solve this linear system is larger than is needed. Further work should focus on allowing Futility to resize its linear solver object dynamically instead of altering the memory or introducing unnecessary computational burden.

2.7 Initial testing and verifcation

In the rewriting of MAMBA3-D into Mongoose, several steps have been taken to best ensure expected consistency between MAMBA3D and the new code. A unit test harness has been included in the re-write with several components having unit tests in place. While not comprehensive in scope, several components have testing in place including water properties, the CRUD_Node, CRUD_Radial and CRUD_Pin routines. A unit test on the crud growth physics also exists since this is in the form of an analytical solution. As the code continues to be developed new unit tests for various pieces of physics will be added. Concurrent with unit testing, code consistency studies with both MAMBA3D and internal consistency studies have been carried out.

Regarding internal consistency, Table 1 shows uniform crud growth occurring on a problem consisting of four axial regions of different heights and four azimuthal regions (which by default in the code will be the same size). Since the initial and boundary conditions in all axial and azimuthal regions are equal, the crud thickness calculation should be identical and they are identical to 10 significant digits. Additional internal consistency checks have taken place throughout development and will be added to the unit test suite as code development continues.

Table 1: Mongoose Internal Consistency Tests

ntheta	dz (cm)	days	crud (microns)
1	1.0	500	28.674
1	2.0	500	28.674
1	3.0	500	28.674
2	1.0	500	28.674
2	2.0	500	28.674
2	3.0	500	28.674
3	1.0	500	28.674
3	2.0	500	28.674
3	3.0	500	28.674
4	1.0	500	28.674
4	2.0	500	28.674

4	3.0	500	28.674

With regards to MAMBA3D consistency with Mongoose, comparisons have been less straightforward. In writing Mongoose, several coding inaccuracies were addressed. For instance, subroutine water property evaluations were performed with the expected input temperature being Kelvin, however Celsius was being passed in. Another issue was found in the internal kinetics routine where a variable passed into kinetic parameter evaluations was being evaluated from a different nodal location than the node being solved. Additional difficulty is encountered because computing the temperature distribution is done a slightly different way. These changes necessarily cause a change in results, thus an expectation of identical answers is not valid. However, some internal parameters are still expected to be identical; in particular items tied to initial or boundary conditions not dependent on the temperature or internal kinetics solve. Thus, items that are expected to be identical are compared to MAMBA3D to check for the consistency. Additionally, while computed internal concentration parameters have changed due to coding adjustments, verifications have been performed to ensure the values are of the same order of magnitude as in MAMBA3D.

Most consistency checks have been performed against the concentration vector which contains both crud surface kinetics and crud internal kinetics. Some of the parameters in the concentration vector are expected to be identical for identical boundary conditions, while others should exhibit similar behavior. In the concentration solution, the eight of the eleven concentrations are calculated in MAMBA3D, with the remaining three being placeholders for future work. These eight concentrations are in units of moles/cm^3 except for porosity which is dimensionless, and consist of the solid nickel ferrite phase, soluble Ni in coolant, soluble Fe in coolant, soluble Boric acid, soluble Li, porosity, soluble H_2, and nickel ferrite particulate concentration which is only used for crud growth. Of these concentrations, all eight are expected to be identical at the beginning of a problem with identical boundary conditions.

Using a sample input deck with 120 radial nodes, a one micron mesh size, one axial zone, three azimuthal zones and identical inputs for temperature, heat flux, and concentration boundary conditions code consistency between MAMBA3D and Mongoose was checked for consistency in the first call to internal kinetics. Shown here in Table 2 identical values at the first call to the internal kinetics solve is seen across all relevant parameters. Subsequent calls do not necessarily give the same answer due to differences in the thermal solution, however initial consistency between the two codes is observed. Again, as consistency checks are performed, the necessary inputs along with the expected outputs will be added to the test suite to ensure any future changes do not adversely affect things.

Table 2: Mongoose Comparisons with MAMBA3D

	MAMBA3D	Mongoose	Relative Difference
$NiFe_2O_4$	6.822E-03	6.822E-03	0.00%
Soluble Fe	2.295E-12	2.295E-12	0.00%
Soluble Ni	5.885E-12	5.885E-12	0.00%
Soluble Boric Acid	5.972E-05	5.972E-05	0.00%
Soluble Li	1.708E-07	1.708E-07	0.00%
Porosity	7.000E-01	7.000E-01	0.00%
Soluble H_2	7.693E-07	7.693E-07	0.00%

To test convergence a mesh sensitivity study was conducted. In this study the sensitivity of the crud growth solution scheme to the mesh node size was evaluated. Varying the node size from 0.0005 microns up to 5.0 microns while also varying the crud growth time step size from 0.001 to 5.0 days, convergence in both spatial and time resolution was determined. For the inputs to the problem under consideration for the study it took approximately 45 days to grow one micron of crud, thus a time step size of 0.1 days and node size of one micron would require 450 time steps to completely fill one crud node. The tradeoff to be determined is how large the node size and time step size can be to still achieve sufficient accuracy.

The larger the node and time step size, the more computationally efficient in both memory and time the problem can be. Table 2 shows the results of the study. The name extensions (5.0, 1.0, 0.1, 0.01 and 0.001) correspond to the time step size used for the crud growth solution. The node thickness at the bottom corresponds to the size of a single node, thus the smaller the node thickness the more refined the mesh is. The comparison selected was the total crud thickness evaluated at 500 days for a representative input deck.

The first item to note is that the 5.0 day time step is noticeably different from the rest of the solutions. Even at a coarse mesh size the crud growth occurs slower in this case than the smaller time step cases. Further, at very small mesh sizes the crud gets significantly smaller, i.e. it does not appear to converge. This occurs since a 5.0 day time step size allows enough crud to accumulate to exceed the mesh size, causing anomalous behavior. At 1.0 day time step size, the result is close to the smaller time step sizes, just 2% different in totally crud thickness at 500 days. Finally, 0.1, 0.01 and 0.001 have nearly identical answers, indicating 1.0 days is sufficient in the time scale.

With respect to node thickness convergence it is clear the mesh is converging near the 1.0 micron thickness. The difference in solution at 0.01 and 0.1 microns is very small compared to 1.0 micron. Beyond one micron the mesh sensitivity shows a clearly increasing slope. Where the difference between 0.01 and 1.0 microns showed a different of ~0.2 microns total crud thickness at 500 days, the different between 1.0 and 5.0 microns was greater than 10 microns. Thus, the recommended mesh size for accuracy and efficiency is 1.0 microns. With respect to crud time step size, 1.0 days will give an adequately accurate answer.

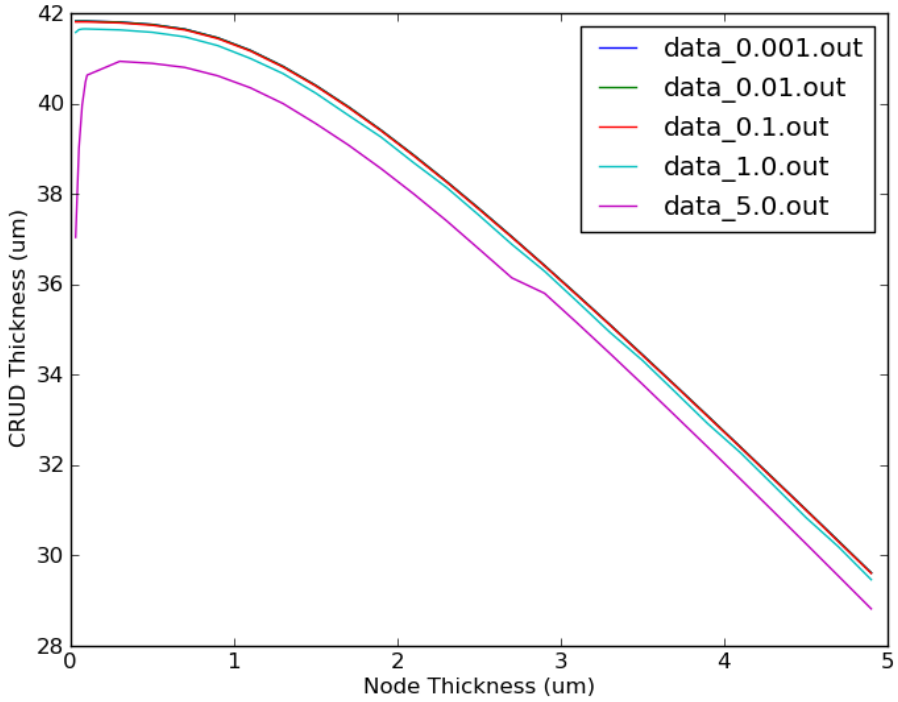


Figure 2: Mongoose Mesh Convergence

3. APPLICATION INTERFACE FOR MONGOOSE

To integrate Mongoose into the rest of VERA, an application programing interface (API) has been developed to allow client codes to control Mongoose behavior. The expected client codes for implementation are expected to be CTF for whole core subchannel analysis and Cicada for localized corrosion analysis. Two different interfaces are exposed; a pin level interface for only Fortran codes and a multipin interface that is accessible from both Fortran and C. Both of these interfaces have very similar capability with the major difference being that the single pin interface requires that the client code perform integrations over the system for system mass balance and for output editing.

The pin based Fortran interface is implemented in the form of a Fortran class which is defined in the module *CRUD_Pin*. The class is then defined as *TYPE(CRUDPin) :: mypin*. The class has several methods to initialize, set data, solve, extract solution data, and clean up the pin. Each of these methods are described below.

Init	
Initialize the internal data structures for the pin	
CALL mypin%init(Ntheta, Nz, hz(:), R_pin, Parameter List)	
	Ntheta – Number of azimuthal divisions around a pin
	Nz – Number of axial divisions in the pin
	hz(:) – The axial height of each axial division (vector length Nz)
	R_pin – Outer radius of the cladding
	Parameter List – Futility parameter list type which contains other options
	nrmax – Maximum number of radial divisions (Default: 200)
	delr – Radial mesh spacing (Default: 1.0E-4 cm)
	tstep_size – Internal time step size (Default: 1 day)
	ODE_tstep_size – Time step size that the fixed step ODE solver takes (Default: 0.01 s)

setCoolantData	
Define the coolant concentrations for the pin	
CALL mypin%setCoolantData(T, $C^p_{NiFe_2O_4}$, C_{Ni}, C_{Fe} , C_B, C_{Li}, V_{H_2}*)*	
	T – Temperature of coolant where concentrations were measured [K]
	$C^p_{NiFe_2O_4}$ – Concentration of particulate nickel ferrite in the coolant [ppb]
	C_{Ni} – Concentration of soluble nickel in the coolant [ppb]
	C_{Fe} – Concentration of soluble iron in the coolant [ppb]
	C_B – Concentration of soluble boron in the coolant [ppm]
	C_{Li} – Concentration of soluble lithium in the coolant [ppm]
	V_{H_2} – Specific volume of hydrogen gas in coolant [cm^3/kg]

setBoundaryData	
Define the thermal and mechanical boundary conditions on each mesh element	
CALL mypin%setBoundaryData(j,k, q'', T_{surf}, T_{cool}, k_{TKE} , [T_sat], [P])	
	j – Azimuthal index for the corresponding boundary data
	k – Axial index for the corresponding boundary data
	q'' – Heat flux entering the crud layer at the outer clad radius [W/cm^2]
	T_{surf} – Temperature on the outer surface of the crud layer [K]
	T_{cool} – Temperature of the bulk coolant surrounding the pin [K]
	k_{TKE} – Local turbulent kinetic energy [J/kg]
	T_{sat} – Local saturation temperature [K] - Optional input (currently unused)
	P – Local pressure [Pa] – Optional input (currently unused)

step	
Performs a time step	
CALL mypin%step(δt)	
	δt – Change in time for the current step [days]

advanceTime	
Updates the time in which the time step starts. The default behavior is to rewind to the previous time step every time *step* is called. *advanceTime* will stop the rewind form occurring.	
CALL mypin%advanceTime( )	
	No input arguments

getCrudData	
Extracts radial integrals of interest for a corresponding azimuthal and axial position	
CALL mypin%getCrudData(j,k, t, $\rho_{NiFe_2O_4}$, ρ_B, R_{th}, $\dot{m}_{evap}$)	
	j – Azimuthal index for the corresponding output data
	k– Axial index for the corresponding output data
	t – thickness of the crud layer [cm]
	$\rho_{NiFe_2O_4}$– Surface density of nickel ferrite [g/cm^2] (density time thickness)
	ρ_B– Surface density of atomic boron [g/cm^2] (both soluble and precipitate)
	R_{th}– Thermal resistance of the crud layer [$K\ cm^2/W$]
	$\dot{m}_{evap}$– Mass evaporation rate [?] (currently unused)

clear	
Deallocate the internal data structures for the pin and return the class to its original state	
CALL mypin%clear( )	
	No input arguments

Future interfaces will include methods to extract data from the pin to support editing 3D solution data and for code restart/fuel shuffling.

The previous interface describes how a Fortran code can directly use the CRUD_pin class to do computations but there are several cases where multiple pins need to be managed inside a distributed computing environment. In order to make this easier, a controller code has been created to manage the mob of Mongoose pins. The controller of this mob is called the DonGoose interface. Many of the interfaces to initialize, set data, control time steps, extract data, and free data are very similar, usually with only an additional index to identify the pin index the user wishes to provide data for.

DonGoose_init	
Initialize the internal data structures for a set of pins	
CALL DonGoose_init(Nlocal, MPI_Comm)	
	Nlocal – Number of pins that exist on this MPI process
	MPI_Comm – The MPI communicator which owns all of the pins

DonGoose_pin_setup	
Initialize the internal data structures for a given pin. It should be noted that the parameter list is not available through this interface because of incompatibility between Fortran and C.	
CALL DonGoose_pin_setup(ipin, irow, jcol, asy, Ntheta, Nz, hz(:), R_pin)	
	ipin – Pin index chosen by client code to represent a pin (between 1 and Nlocal)
	irow – The row index inside an assembly which this pin represents (for output edits)
	jcol – The column index inside an assembly which this pin represents (for output edits)
	asy – The assembly index in the core which this pin represents (for output edits)
	Ntheta – Number of azimuthal divisions around a pin
	Nz – Number of axial divisions in the pin
	hz(:) – The axial height of each axial division (vector length Nz)
	R_pin – Outer radius of the cladding

DonGoose_setCoolantData	
Define the coolant concentrations for all pins	
CALL DonGoose_setCoolantData(T, $C^p_{NiFe_2O_4}$, C_{Ni}, C_{Fe} , C_B, C_{Li}, V_{H_2}*)*	
	T – Temperature of coolant where concentrations were measured [K]
	$C^p_{NiFe_2O_4}$ – Concentration of particulate nickel ferrite in the coolant [ppb]
	C_{Ni} – Concentration of soluble nickel in the coolant [ppb]
	C_{Fe} – Concentration of soluble iron in the coolant [ppb]
	C_B – Concentration of soluble boron in the coolant [ppm]
	C_{Li} – Concentration of soluble lithium in the coolant [ppm]
	V_{H_2} – Specific volume of hydrogen gas in coolant [cm^3/kg]

DonGoose_pin_setBoundaryData	
Define the thermal and mechanical boundary conditions on each mesh element	
CALL DonGoose_pin_setBoundaryData(ipin,j,k, q'', T_{surf}, T_{cool}, k_{TKE} , [T_sat], [P])	
	$ipin$ – Pin index for the corresponding boundary data
	j – Azimuthal index for the corresponding boundary data
	k – Axial index for the corresponding boundary data
	q'' – Heat flux entering the crud layer at the outer clad radius [W/cm^2]
	T_{surf} – Temperature on the outer surface of the crud layer [K]
	T_{cool} – Temperature of the bulk coolant surrounding the pin [K]
	k_{TKE} – Local turbulent kinetic energy [J/kg]
	T_{sat} – Local saturation temperature [K] - Optional input (currently unused)
	P – Local pressure [Pa] – Optional input (currently unused)

DonGoose_step	
Performs a time step for all pins	
CALL DonGoose_step(δt)	
	δt – Change in time for the current step [days]

DonGoose_advanceTime	
Updates the time in which the time step starts for all pins. The default behavior is to rewind to the previous time step every time *step* is called. *advanceTime* will stop the rewind form occurring.	
CALL DonGoose_advanceTime( )	
	No input arguments

DonGoose_pin_getCrudData	
Extracts radial integrals of interest for a corresponding pin, azimuthal, and axial position	
CALL DonGoose_pin_getCrudData(ipin,j,k, t, $\rho_{NiFe_2O_4}$, ρ_B, R_{th}, $\dot{m}_{evap}$)	
	$ipin$ – Pin index for corresponding output data
	j – Azimuthal index for the corresponding output data
	k– Axial index for the corresponding output data
	t – thickness of the crud layer [cm]
	$\rho_{NiFe_2O_4}$– Surface density of nickel ferrite [g/cm^2] (density time thickness)
	ρ_B– Surface density of atomic boron [g/cm^2] (both soluble and precipitate)
	R_{th}– Thermal resistance of the crud layer [$K\ cm^2/W$]
	$\dot{m}_{evap}$– Mass evaporation rate [?] (currently unused)

DonGoose_clear	
Deallocate the internal data structures for all pins and resets state	
CALL DonGoose_clear( )	
	No input arguments

In addition to these interfaces, a Fortran specific interface is added to allow Fortran codes to use DonGoose to manage the multipin components but interact with the pin object also. This routine provides the client code with a pointer to the *CRUD_Pin* object stored internally by DonGoose.

DonGoose_getPinPtr	
Sets pointer to DonGoose internally allocated CRUD_Pin class. Should be used in place of DonGoose_pin_setup. Once the pointer is obtained, client code should directly call Init on the object. All other operations can be performed using the pointer or with DonGoose interface.	
CALL DonGoose_getPinPtr(ipin, irow, jcol, asy, CRUD_Pin_Ptr)	
	ipin – Pin index chosen by client code to represent a pin (between 1 and Nlocal)
	irow – The row index inside an assembly which this pin represents (for output edits)
	jcol – The column index inside an assembly which this pin represents (for output edits)
	asy – The assembly index in the core which this pin represents (for output edits)
	CRUD_Pin_Ptr – A pointer of type CRUD_Pin which DonGoose will assign appropriately

4. COUPLING CONSIDERATIONS

The previous coupling in VERA between MPACT, CTF, and MAMBA1D will be highly leveraged when integrating in Mongoose. There are however, some suggestions on improving this interface. To provide context for the changes, first a high-level overview of the previous coupling will be discussed.

In previous work, a crud interface was developed in CTF which would call a MAMBA1D instance on each quadrant of the pin and at every axial level. CTF explicitly meshes the crud layer in its heat conduction solve and uses a thermal resistance provided by MAMBA1D. CTF will provide MAMBA1D the heat flux on the outside of the cladding and the surface temperature on the outside of the crud layer which is then used to predict the crud growth, boron precipitate, and internal temperature of the crud layer.

CTF will then take the crud thickness, crud mass, and boron mass and azimuthally integrate over the pin before passing to MPACT. MPACT homogenizes the nickel ferrite and lithium tetraborate into a user defined layer (typically 150-200 microns) on the outside of the cladding. This allows MPACT to capture the suppression of the flux locally and is reflected in the power shape which will be provided to CTF.

The iteration scheme between MPACT, CTF, and MAMBA1D occurs until the power shape in the core is sufficiently converged. Once this occurs, the next time step can be taken. Along with the major physics components, a system mass balance capability is provided in CTF. This looks at the concentration of nickel ferrite particulate in the coolant. There is a source term from the steam generators and a sync from the growth of crud in the core. While all of these components are capable of representing the physics, there are a few improvements that Mongoose can provide into this framework.

System Mass Balance

First, a tighter coupling with the mass balance can be made. Each pin can explicitly track how much particulate nickel ferrite (and eventually nickel oxide) that is captured on the surface of the crud. This would be in contrast to other soluble species that could be removed from the coolant because of precipitation. Mongoose could provide an interface to provide the change in the

various coolant species absorbed into the crud layer. The DonGoose interface could also expose this method which would be integrated over the entire core.

Improving coupling with CTF and Cicada

Currently the thermal resistance that is used to mimic the temperature change across the crud layer is insufficient to describe the complex heat transfer that occurs in the crud layer. Because a major contributor to the heat transfer inside the crud is boiling inside the crud chimneys, there is a significant component of the heat which is directly transferred from inside the crud layer into the bulk coolant through water vapor. Only a fraction of the heat flux which enters the crud layer at the cladding, will need to be conducted through the entire crud layer and need to be transferred to the coolant by convection,

$$\boldsymbol{q''_{cool}A_{cool} = q''_{clad}A_{clad} - \dot{q}_{boil}V_{crud} = hA(T - T_{cool})}, \quad [25]$$

where $\dot{q}_{boil}$ is the volumetric heat transfer caused by boiling. By not accounting for the heat transfer from boiling, the outer surface temperature of the crud will be over predicted. While the thermal heat resistance can give the correct temperature change across the crud layer, it is insufficient to account for the energy transfer that occurs in the crud layer. A mass evaporation rate was also previously provided by MAMBA1D which could be used to account for this effect. The issue with this term is it depends on consistent water properties, specifically the latent heat of vaporization to obtain a consistent heat removal.

Instead, it is proposed that Mongoose expose the heat transfer solution in the crud layer to the client code. This way, CTF and Cicada can directly model the heat transfer through the crud layer as part of their conjugate heat transfer solution. Because the growth of the crud layer is also tightly coupled to the boiling rates in the crud, an iterative strategy will be required to resolve the time dependent growth and densification of nickel ferrite and lithium tetraborate and the temperature dependence. The coupling with the thermal solve is to allow CTF and Cicada to converge to the correct heat flux and surface temperature more quickly during this iterative strategy.

Improvements to MPACT Interface

The current MPACT interface assumes only nickel ferrite and lithium tetraborate components in the crud layer. The current interface also must make assumptions about the water density inside the pores in the crud layer. In order to better capture the composition of species inside the crud layer, the Mongoose interface should be extended to directly provide number densities of the elements.

Likewise, since boron exists in multiple forms, soluble and in precipitate. It makes sense for Mongoose to directly model the B-10 depletion. The soluble boron is expected to continually refresh with the coolant boron so the depletion effects are minimal. But once the boron is entrapped in the precipitate, it is expected to deplete. Currently the model exists in MPACT is based on depleting all the boron mass provided by MAMBA1D but scaled by a factor provided by the user. While this factor may still be needed, only depleting the precipitate boron should provide a more stable model. This would require MPACT to provide the microscopic reaction rate of B-10 inside the crud layer and the current abundance of B-10 in the coolant.

The three modifications suggested above would ensure consistent physics between MPACT, CTF, and Mongoose and Cicada and Mongoose. Each component would require some work to modify the interface and in the corresponding physics codes to compute the extra data needed

but it is believed that it would enhance the ability of the coupled code system to accurately model CIPS.

5. CONCLUSIONS

This report documents the methods and models that have been developed into the Mongoose coolant chemistry code in support of the CASL crud challenge problems (CIPS and CILC). It also discusses the APIs which are exposed to both Fortran and C based client codes. Lastly, a discussion of potential modifications to the existing coupling interfaces were made to improve consistency between the various physics codes.

The path forward to support the CIPS challenge problem involves several key steps. First, the Mongoose interface needs to be fully integrated and tested in CTF, including integration with the source term calculation. Once this is complete, the integration with MPACT should be identical to MAMBA1D. A demonstration of Watts Bar Unit 1 Cycle 7 will be made to assess the current speed and any limitations of the code. Concurrently, calibration of the models in Mongoose to existing measured data will occur. Because many of the models require experimental data to determine the coefficients, calibration of the code against existing data sets will be needed. One of the parameters which currently does not have good experimental data is the precipitation of lithium tetraborate. The axial offset caused by CIPS in Watts Bar Unit 1 Cycle 7 may be needed to calibrate the precipitation model. The final runs will test 3 different fuel cycles; cycle 6 which shows mild CIPS, cycle 7 which shows significant CIPS, and cycle 8 which has no CIPS. A successful run of these 3 cycles would give confidence that the physics in Mongoose is capable of distinguishing between cycles with and without CIPS.

REFERENCES

1. B. Kendrick. MAMBA Theory Manual. Los Alamos National Laboratory. 2016
2. B. Collins, et.al. Simulation of Crud-Induced Power Shift using the VERA Core Simulator and MAMBA. Physor 2016. 2016.
3. Futility Development Group. “Futility – FORTRAN Utility” Revision 2.1.0. https://github.com/CASL/Futility.
4. A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "*SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers*," ACM Transactions on Mathematical Software, 31(3), pp. 363-396, 2005.