Numerical Coupling Interface in Rocstar

Rocman (GEN 3) Design Document (Draft)

P. H. Geubelle X. Jiao A. C. Haselbacher M. T. Campbell

July 9th, 2004

1 Introduction

The purpose of this document is to describe in detail the coupling algorithm and information transfer between the various components (fluid, solid, combustion, interface, and manager) of the GEN3 of the Rocstar integrated code. This document also serves as the Design Document for Rocman.

The main differences between GEN3 and GEN2 are the use of Rocin and Rocout for I/O, allowing Rocburn to reside on either fluid or solid surface, and enforcement of conservation of energy. In GEN 3, the formulation of time-zooming (F. Najjar and R. Moser) has also been incorporated.

1.1 General notation

S = solid code
F = fluid code
B = combustion (burn) code, operates on burning surface patches of S or F
I = interface code
M = manager code
NB = non-burning surface panes (patches) of S or F
NI = non-interacting surface panes (patches) of S or F

$\boldsymbol{u}$ = displacement field
$\boldsymbol{v}$ = velocity field
$\boldsymbol{a}$ = acceleration field
$\boldsymbol{t}$ = traction field
$\dot{m}$ = mass flux
$\dot{r}_b$ = burning rate
ρ = density
p = pressure
q = heat flux
S =heat release due to chemical reactions
T = temperature
$\boldsymbol{x}$ = undeformed location
$\boldsymbol{y}$ = deformed location
$\boldsymbol{n}$ = surface normals (deformed; measured positive into solids)
$\boldsymbol{N}$ = surface normals (undeformed; measured positive into solids)

$(\cdot)_{SN}$= value computed at nodes of the solid interface
$(\cdot)_{SF}$= value computed at faces of the solid interface

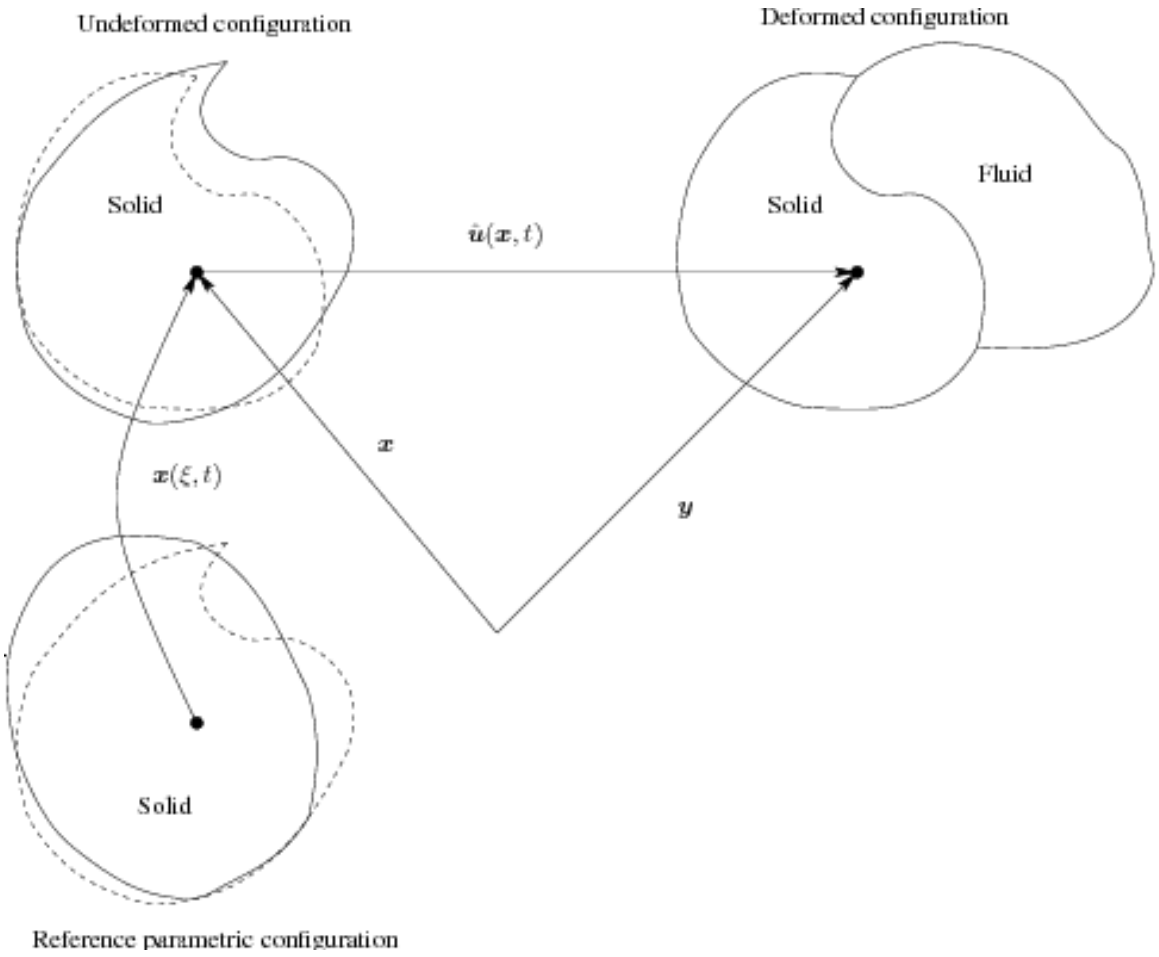

Figure 1: Schematic overview of reference, undeformed, and deformed configurations as used in the solid code.

$(\cdot)_{FN}$= value computed at nodes of the fluid interface
$(\cdot)_{FF}$ = value computed at faces of the fluid interface
$(\cdot)_{BF}$ = value computed at faces of the burning interface (fluid or solid)
$(\cdot)_{BN}$ = value computed at nodes of the burning interface (fluid or solid)
$(\hat{\cdot})$ = value associated with particles
$(\bar{\cdot})$ = value associated with mesh

iPredCorr = step index of predictor-corrector iterations (starting with 1)
solids_burn = combustion operates on solid mesh
fluids_burn = combustion operates on fluid mesh
solids_ALE = surface motion due to burning is on in solids

The deformed and undeformed solid domains are depicted schematically in Figure 1.

1.2 Jump conditions at fluid/structure interface

1.2.1 ALE formulation

This refers to the case where surface regression due to burning (i.e., solids_ALE) is enabled.

Conservation of mass: $\boldsymbol{v}_f \cdot \boldsymbol{n} = -\dfrac{\dot{m}}{\rho_f} + (\boldsymbol{v}_s + \dot{r}_b \boldsymbol{n}) \cdot \boldsymbol{n}$ [Here $\boldsymbol{n}$ is incoming to solids.]

No slip: $\boldsymbol{v}_f \cdot \hat{\boldsymbol{t}} = \boldsymbol{v}_s \cdot \hat{\boldsymbol{t}}$, where $\hat{\boldsymbol{t}}$ denotes surface tangents.

The above two conditions reduce to $\boldsymbol{v}_f = \boldsymbol{v}_s + \left(\dot{r}_b - \dfrac{\dot{m}}{\rho_f}\right)\boldsymbol{n}$.

Conservation of momentum: $\boldsymbol{t}_s = \boldsymbol{t}_f + \dot{m}(\boldsymbol{v}_s - \boldsymbol{v}_f) = \boldsymbol{t}_f + \dot{m}\left(\dot{r}_b - \frac{\dot{m}}{\rho_f}\right)\boldsymbol{n}$

Conservation of energy: $T \equiv T_f = T_s$ and $q_s \approx q_f + \dot{m}Q + S$,
where $q_s \equiv \boldsymbol{q}_s \boldsymbol{n}$, $q_f \equiv \boldsymbol{q}_f \boldsymbol{n}$, $Q \equiv T(C_s - C_{f,v})$, T is the common interface temperature, and S is the heat release due to chemical reactions. The latter equality is obtained by ignoring negligible terms in

$$q_s = q_f + \dot{m}T(C_s - C_{f,v}) + \frac{1}{2}\dot{m}(\boldsymbol{v}_s^T \boldsymbol{v}_s - \boldsymbol{v}_f^T \boldsymbol{v}_f) + \boldsymbol{t}_s^T \boldsymbol{v}_s - \boldsymbol{t}_f^T \boldsymbol{v}_f + S,$$

where C_s is the heat capacity constant of the solid, and $C_{f,v}$ is the specific heat at constant volume in the fluid.

Continuity of displacements: $\boldsymbol{u}_f = \boldsymbol{u}_s$.

1.2.2 No-ALE formulation

This refers to the case where surface regression (i.e., solids_ALE) is disabled by imposing $\dot{r}_b = 0$ in the conditions.

Conservation of mass: $\boldsymbol{v}_f \cdot \boldsymbol{n} = -\dfrac{\dot{m}}{\rho_f} + \boldsymbol{v}_s \cdot \boldsymbol{n}$ [Here $\boldsymbol{n}$ is incoming to solids.]

No slip: $\boldsymbol{v}_f \cdot \hat{\boldsymbol{t}} = \boldsymbol{v}_s \cdot \hat{\boldsymbol{t}}$, where $\hat{\boldsymbol{t}}$ denotes surface tangents.

The above two conditions reduce to $\boldsymbol{v}_f = \boldsymbol{v}_s - \dfrac{\dot{m}}{\rho_f}\boldsymbol{n}$.

Conservation of momentum: $\boldsymbol{t}_s = \boldsymbol{t}_f + \dot{m}(\boldsymbol{v}_s - \boldsymbol{v}_f) = \boldsymbol{t}_f - \frac{\dot{m}^2}{\rho_f}\boldsymbol{n}$.

Jump conditions for T_f, q_s, and $\boldsymbol{u}_f$ remain the same.

1.2.3 Time zooming

Rocman incorporates the time-zooming formulation by Fady Najjar and Bob Moser. As a result, a time-zoom factor z is introduced as a parameter in Rocman, so that the burning rate under time-zooming, denoted by $\dot{r}_b^z$, would equal to $z\dot{r}_b$ under the same system time step. Assuming the interface condition has achieved a quasi-steady state, the effect of time zooming is that the interface regresses z times as far as the normal burning during the system time step, while maintaining the quasi-steady state of the interface in terms of $\rho_f(\boldsymbol{v}_s - \boldsymbol{v}_f)$, which remains the same with or without time zooming. Let $\dot{m}^z$ denote the mass flux under time zooming and $\dot{m}$ denote the normal mass flux. From conservation of mass and the no-slip conditions, we obtain $\dot{m}^z - z\rho_f \dot{r}_b = \dot{m} - \rho_f \dot{r}_b$, and hence

$$\dot{m}^z = \dot{m} + (z-1)\rho_f \dot{r}_b.$$

Furthermore, since the total mass is no longer conserved under time zooming, we use the simple calculation of $\dot{m} = \rho_s \dot{r}_b$, which leads to $\dot{m}^z = \dot{m}\left(1 + \rho_f/\rho_s(z-1)\right)$. In the case of No-ALE formulation, this reduces to $\dot{m}^z = \dot{m}$.

From a user's perspective, time zooming has the effect of advancing the geometry of the interface for a time period of $z\Delta t$, and hence Rocman advances the time stamp by $z\Delta t$. Note that when $z = 0$, Rocman will reset z to 1 for all computations except that the surface motion will be disabled.

[Note: Rocman currently *does not* implement the zoomed mass flux. The fluid code implements time zooming in such a way that the nominal mass flux should be used instead of the zoomed mass flux. In coupled runs, mass flux is computed geometrically as the volume swept out by the propagating face. When zooming is used in coupled runs, the physics modules are responsible for dealing with the increase over the nominal mass flux values.]

[Note: For the evaluation of $\boldsymbol{t}_s = \boldsymbol{t}_f + \dot{m}(\boldsymbol{v}_s - \boldsymbol{v}_f)$, it is not yet clear whether $\dot{m}$ or $\dot{m}^z$ should be used and will be investigated later.]

1.3 Subcycling scheme

The individual component codes (F, S and B) proceed at their respective time steps (Δt_F, Δt_S, and Δt_B), i.e., they are allowed to subcycle based on their respective stability (Courant) and precision criteria. However, information is transferred between these component codes only at every "system time step" Δt, which is greater or equal to the largest of the three "component time steps".

1.3.1 Interpolation and extrapolation

During subcycling, a physical component can request boundary conditions from Rocman at any time $t \in [t^n, t^{n+1}]$ using a local time index $\alpha = (t - t^n)/\Delta t^n \in [0...1]$. Rocman interpolates or extrapolates values using the following schemes to obtain values at time $n + \alpha$. First, if it has values at times n and $n + 1$, the interpolation takes the form

$$v^{n+\alpha} = v^n + \alpha(v^{n+1} - v^n).$$

Second, if has values at time $n - 1$ and n, and the extrapolation takes the form

$$v^{n+\alpha} = v^n + \alpha\Delta t^n \frac{v^n - v^{n-1}}{\Delta t^{n-1}}.$$

Note that we may also simplify them and have $v^{n+\alpha} = v^n$. Using interpolated/extrapolated values might deliver higher-order of accuracy and allow use of larger system time steps (i.e., to have more subcycles).

1.3.2 Subcycling without Predictor-Corrector Iterations

Figure 2 shows a schematic overview of the coupling among the physical modules as they march in time from t^n to t^{n+1} without predictor-corrector iterations. The left-hand side of the figure indicates how the three physical modules march at their own time steps to meet the user-specified system time step. The right-hand side of the figure depicts variations of some of the interface quantities in time. Interface quantities associated with the combustion code are shown in red, a solid interface quantity is shown in green, and some fluid interface variables are indicated in blue.

At the beginning of a system time step, the combustion code takes as many steps as required to meet the specified system time step. As the fluid code has not yet marched its solution from t^n to t^{n+1}, the combustion code receives time-extrapolated values for the fluid temperature and heat flux at the interface from the manager. The extrapolated variations are indicated by dashed blue lines, and the values passed to the combustion code are marked by the cross symbols.

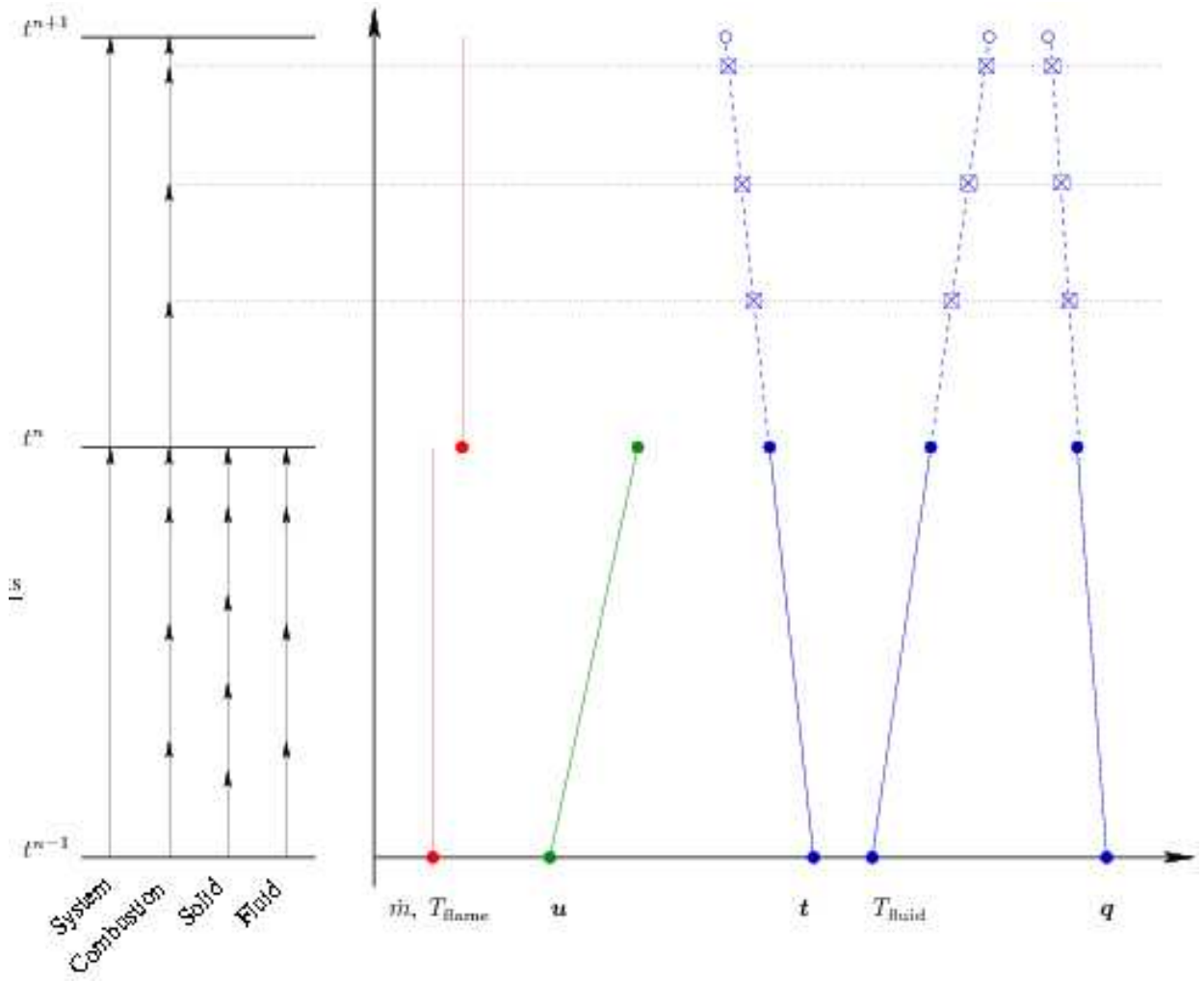

Figure 2: Coupling without predictor-corrector steps.

The extraction of the combustion code from the fluid code and the use of extrapolated values in the coupling algorithm leads to two important benefits: First, it increases the modularity of the coupled code and hence the ease with which plug-and-play may be performed. Second, and perhaps more important, the variation in time of each interface quantity is now uniquely defined by the manager. This means that it should be possible to enforce rigorously mass conservation even when a simulation employs subcycling. (It should be noted that the current version of GEN3 was only tested for mass conservation without subcycling, precisely because it was not straightforward to define a unique mass flux as a function of time.)

As indicated in the figure, we propose that—at least in a first implementation—the injection mass flux and temperature computed by the combustion code are assumed to be constant over a system time step.

1.3.3 Subcycling with Predictor-Corrector Iterations

A schematic overview of the coupling among the physical modules as they march in time from t^n to t^{n+1} with predictor-corrector steps is shown in Fig. 3. In contrast to the previous figure, dashed lines indicate variations of interface quantities during previous predictor-corrector steps. Thus the manager can provide, for example, the combustion code with a fluid temperature and heat flux at the interface based on the varia-

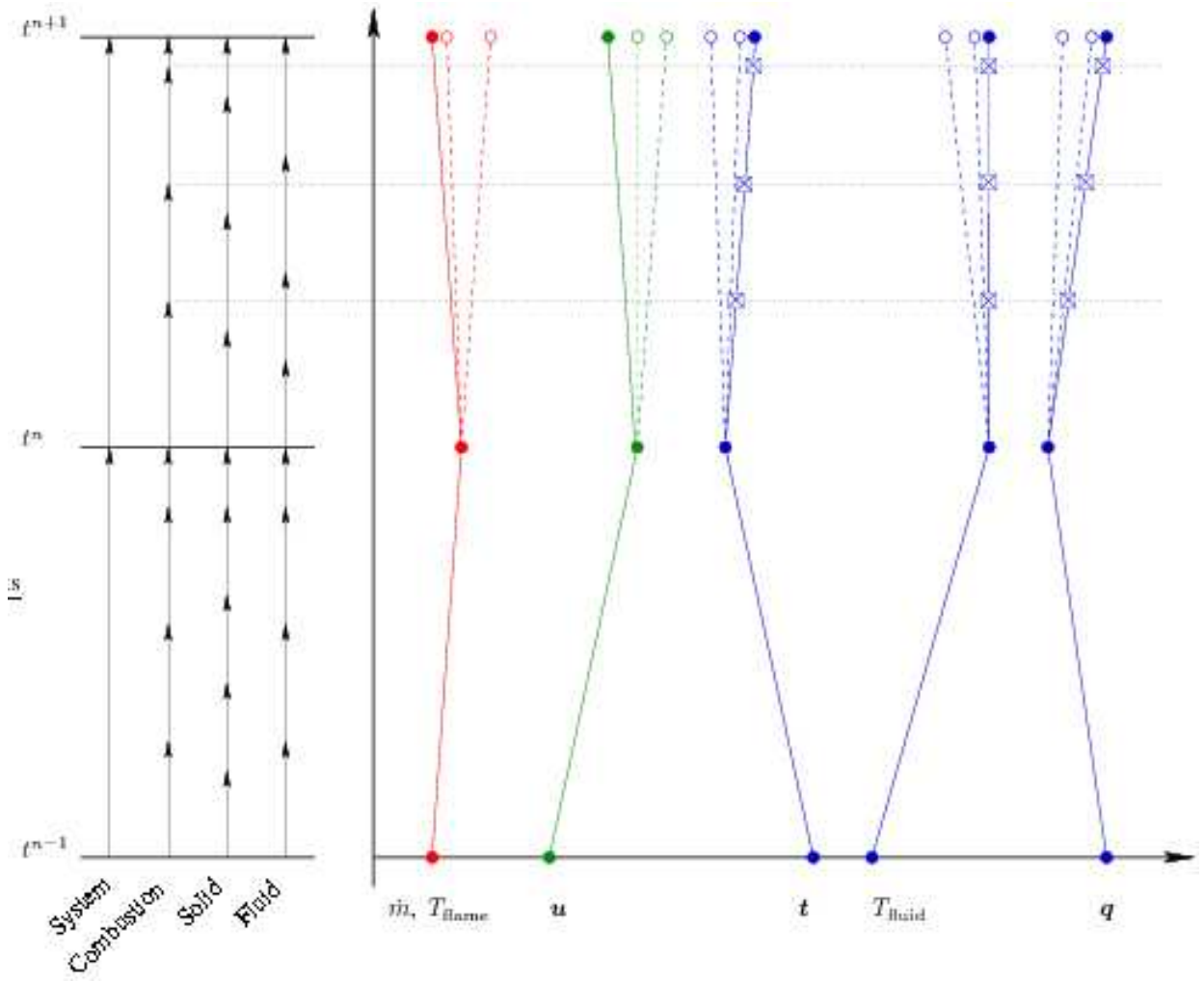

Figure 3: Coupling with predictor-corrector steps.

tion of these quantities from the last predictor-corrector step. Only during the first predictor-corrector step is it necessary to use extrapolation.

1.4 Communication mechanism

All components communicate with each other through Roccom and Rocman. Each component owns some buffers for storing incoming or outgoing messages, which are registered to Roccom. Rocman copies data from outgoing buffers to respective incoming buffers, and performs algebraic manipulations for the buffered data as necessary. In this documentation, "send" indicates putting data into outgoing buffers. Rocman provides subroutines to manipulate interface data as described later. The prerequisite of these subroutines are that the data in outgoing buffers are up-to-date. The output of these subroutines are stored in incoming buffers of the physical components.

2 Interface Buffer Data

The physical modules need to provide storage for the interface quantities listed below. Interface quantities can vary according to certain runtime options.

2.1 Solid domain

$\hat{\boldsymbol{u}}(\xi,t) = \boldsymbol{u}(\boldsymbol{x}(\xi,t),t) =$ particle displacement
$\hat{\boldsymbol{v}}(\xi,t) = \partial\hat{\boldsymbol{u}}/\partial t(\xi,t) =$ parameterized velocity
$\hat{\boldsymbol{a}}(\xi,t) = \partial^2\hat{\boldsymbol{u}}/\partial t^2(\xi,t) =$ parameterized acceleration
$\bar{\boldsymbol{u}}(\xi,t) = \boldsymbol{x}(\xi,t) - \boldsymbol{x}(\xi,0) =$ mesh displacement
$\bar{\boldsymbol{v}}(\xi,t) = \partial\boldsymbol{x}/\partial t(\xi,t) =$ mesh velocity
$\bar{\boldsymbol{a}}(\xi,t) = \partial^2\boldsymbol{x}/\partial t^2(\xi,t) =$ mesh acceleration
$\boldsymbol{v}(\boldsymbol{x},t) = \partial\boldsymbol{u}/\partial t(\boldsymbol{x},t) = \hat{\boldsymbol{v}} - \nabla_x\boldsymbol{u}\cdot\bar{\boldsymbol{v}} =$ particle velocity
$\boldsymbol{y}(\xi,t) = \boldsymbol{x}(\xi,t) + \hat{\boldsymbol{u}}(\xi,t) =$ deformed location

Outgoing data

$(C_s)_{SF}^{n+1}$ [New for GEN3], $(\rho_s)_{SF}^{n+1}$, $(T_s)_{SN}^{n+1}$ [New for GEN3], $(\boldsymbol{v}_s)_{SN}^{n+1}$, $(\hat{\boldsymbol{u}})_{SN}^{n+1} = (\boldsymbol{y}-\boldsymbol{x})_{SN}^{n+1}$, $(\Delta\boldsymbol{u})_{SN}^{n+1}$, [defined on interacting surface patches]
and $(\boldsymbol{x})_{SN}^{n+1}$ [defined on both interacting and noninteracting patches]
bcflag : a pane attribute indicating burning (=1), nonburning (=0), and noninteracting (=2) [New for GEN3.]

Incoming data

$(\boldsymbol{t}_s)_{SF}^{n+\alpha}$ and $(q_s)_{SF}^{n+1}$ [New for GEN3, defined on interacting surface patches]
$(\bar{\boldsymbol{v}})_{SN}^{n+\alpha}$, [defined on both interacting and noninteracting surface patches]
$(\text{bflag})_{BF}^{n+1}$, [defined only if solids_burn, and defined on burning patches]

[Remark: If solids_burn, $(\text{bflag})_{BF}^{n+1}$ comes from B and indicates whether a face has been ignited. During initialization, $(\text{bflag})_{BF}^{n+1}$ is outgoing from S to B.]

2.2 Fluid domain

Outgoing data

$(C_{f,v})_{FF}^{n+1}$[New for GEN3],$(p_f)_{FF}^{n+1}$, $(q_c)_{FF}^{n+1}$ (convective heat flux), $(q_r)_{FF}^{n+1}$ (radiation heat flux), $(\boldsymbol{n}_f)_{FF}^{n+\alpha}$ (outwards normal), $(\rho_f)_{FF}^{n+\alpha}$, $(\boldsymbol{t}_f)_{FF}^{n+1}$, $(T_{fout})_{FF}^{n+1}$, [defined on interacting surface patches]
$(\boldsymbol{y})_{FN}^{n+1}$,[defined on both interacting and noninteracting surface patches], and $(bflag)_{FF}^{n+1}$[defined on interacting surface patches]
bcflag , a pane attribute indicating burning (=1), nonburning (=0), and noninteracting (=2)

[New changes to GEN3: T_f is renamed to T_{fout}.]

Incoming data

$(\dot{m})_{FF}^{n+\alpha}$, $(T_{fin})_{FF}^{n+\alpha}$, $(\Delta\boldsymbol{u})_{FN}^{n+\alpha}$ (nodal displacement vector of grid points due to boundary motion), $(\rho_f\boldsymbol{v}_f)_{FF}^{n+\alpha}$, [defined on interacting surface patches], and $(\text{bflag})_{BF}^{n+1}$ [(incoming during ignition modeling) defined only if fluids_burn, and defined on burning patches]

[Remark: If fluids_burn and there is ignition modeling, then$(\text{bflag})_{BF}^{n+1}$ comes from B and indicates whether a face has been ignited. At all other times, $(\text{bflag})_{BF}^{n+1}$ is outgoing from F, and indicates whether a face is burning.]

[New changes to GEN3: T_{flame} is renamed to T_{fin}.]

2.3 Combustion

Outgoing data

$(\dot{r}_b)_{BF}^{n+1}$, $(S)_{BF}^{n+1}$ [New for GEN3], $(T_{bout})_{BF}^{n+1}$, and $(\text{bflag})_{BF}^{n+1}$

[Remark: Depending on fluids_burn or solids_burn, B refers to the burning patches of F or S, respectively. Before ignition, T_{bout} is surface temperature. $(\text{bflag})_{BF}^{n+1}$ comes from B and indicates whether a face has been ignited. During initialization, and after full ignition, $(\text{bflag})_{BF}^{n+1}$ is outgoing from F or S to B, depending on whether fluids_burn or solids_burn is true.]

Incoming data

$(p_f)_{BF}^{n+\alpha}$, $(q_c)_{BF}^{n+\alpha}$, $(q_r)_{BF}^{n+\alpha}$, $(\rho_s)_{BF}^{n+\alpha}$, $(T_{bin})_{BF}^{n+\alpha}$,
and $(\boldsymbol{y})_{BF}^{n}$ [for film-coefficient calculation]

2.4 Manager

Intermediate data for combustion

$(p_f)_{BF}^{-}$, $(q_c)_{BF}^{-}$, $(q_r)_{BF}^{-}$, $(\rho_s)_{BF}^{-}$, and $(T_{bin})_{BF}^{-}$, [defined only on burning patches]
and $(\rho_s)_{BF}^{n+1}$ [defined on all interacting boundaries]

Intermediate data for solids

$(\bar{\boldsymbol{v}}_s)_{SN}^{n+1}$, $(\bar{\boldsymbol{v}}_s)_{SN}^{-}$, $(\boldsymbol{N})_{SF}^{n+1}$, [for surface propagation, defined on interacting and noninteracting patches]

$(A)_{SF}^{n+1}$, $(\dot{m})_{SF}^{n+1}$, $(\dot{r}_b)_{SF}^{n+1}$ [for computation of mass flux], $(q_c)_{SF}^{n+1}$, $(q_r)_{SF}^{n+1}$ and $(q_s)_{SF}^{-}$, [New for GEN3], $(C_{f,v})_{SF}^{n+1}$[New for GEN3], $(S)_{SF}^{n+1}$ [New for GEN3], $(T_s)_{SF}^{n+1}$ [New for GEN3], $(\boldsymbol{t}_s)_{SF}^{n+1}$, $(\boldsymbol{t}_s)_{SF}^{-}$, $(\boldsymbol{u})_{SN}^{n+1}$, $(\boldsymbol{v}_s)_{SF}^{n+1}$, and $(\boldsymbol{y})_{SN}^{n}$ [defined on interacting patches]

Intermediate data for fluids

$(\dot{m})_{FF}^{n+1}$, $(\dot{m})_{FF}^{-}$, $(\dot{r}_b)_{FF}^{n+1}$ [for solids_burn], $(\dot{r}_b)_{FF}^{-}$ [either fluids_burn or solids_burn], $(\dot{r}_b)_{FF}^{n+\alpha}$, $(\dot{r}_b)_{FF}^{-}$, $(q_f)_{FF}^{n+1}$[New for GEN3], $(S)_{FF}^{n+1}$ [New for GEN3, if fluids_burn], $(\boldsymbol{t}_s)_{FF}^{n+1}$, $(\boldsymbol{v}_s)_{FF}^{n+1}$, $(\boldsymbol{v}_s)_{FF}^{n+\alpha}$, $(\boldsymbol{v}_s)_{FF}^{-}$,$(T_{fin})_{FF}^{-}$ [New for GEN3],

$(\Delta\boldsymbol{u})_{FN}^{n+1}$, $(\boldsymbol{v}_m)_{FN}^{n+1}$, $(\boldsymbol{v}_m)_{FN}^{n+\alpha}$, $(\boldsymbol{v}_m)_{FN}^{-}$, $(\boldsymbol{y})_{FN}^{0}$, [helpers for computing $(\Delta\boldsymbol{u})_{FN}$ or transferring from S to F (See Section 3.6)]

$(\dot{m})_{FF}^{\text{pre}}$, $(\boldsymbol{t}_s)_{FF}^{\text{pre}}$, $(\boldsymbol{v_s})_{FF}^{\text{pre}}$, and $(\Delta\boldsymbol{u})_{FF}^{\text{pre}}$ [for convergence check if with PC iterations].

[Remark: In the above, the ones with time levels '$-$' store solutions of the previous system time step for extrapolation or interpolation.]

Control parameters

Data transfer parameters: mode of load transfer (pressure or tractions), order of interpolation in time
z: time-zooming factor
ρ_s: solid density for fluid-alone simulations

3 Control flow

This section describes the control flow of the fully-coupled code. Rocman also provides support for running in fluid- and solid-alone mode, and the steps/substeps needed in the two modes are marked by FA and SA at the right margin, respectively. The substeps related to time zooming are marked by a Z at the right margin. The substeps that depend on ALE and No-ALE are marked by ALE.

Starting point: assume the solution is known everywhere in the fluid and solid domains at time t^n.

0. Initialization (to be called only at time 0.)

   (a) F sends $(C_{f,v})^0_{FF}$, $(T_{fout})^0_{FF}$, $(q_c)^0_{FF}$, $(q_r)^0_{FF}$, $(p_f)^0_{FF}$, $(\rho_f)^0_{FF}$, $(\boldsymbol{t}_f)^0_{FF}$, $(\boldsymbol{y})^0_{FN}$, $(\boldsymbol{n}_f)^0_{FF}$, and $(\text{bflag})^0_{FF}$ to M — FA

   (b) S sends $(C_s)^0_{SF}$, $(T_s)^0_{SN}$, $(\rho_s)^0_{SF}$, $(\hat{\boldsymbol{u}})^0_{SN}$, and $(\boldsymbol{x})^0_{SN}$ to M — SA

   (c) M computes $(\boldsymbol{y})^0_{SN} = (\boldsymbol{x})^0_{SN} + (\hat{\boldsymbol{u}})^0_{SN}$

   (d) If (fluids_burn), then M transfers ρ_s to $(\rho_s)^0_{FF}$ and $(T_{fout})^0_{FF}$ to $(T_{bin})^0_{BF}$, — FA
   else (solids_burn), then M transfers $(q_c)^0_{FF}$, $(q_r)^0_{FF}$, and $(p_f)^0_{FF}$ to S and $(T_s)^0_{SF}$ to $(T_{bin})^0_{BF}$

   (e) B obtains $(T_{bin})^0_{BF}$, $(q_c)^0_{BF}$, $(q_r)^0_{BF}$, $(p_f)^0_{BF}$, and $(\rho_s)^0_{BF}$ from M and sends $(\dot{r}_b)^0_{BF}$ to M — FA

1. Manager: (init_inbuff for B)

   (a) If (fluids_burn), then M transfers $(\rho_s)_{SF}$ to $(\rho_s)_{FF}$, $(T_{fout})^n_{FF}$ to $(T_{bin})^n_{BF}$, and p_f, q_c, q_r from FF to BF
   else (solids_burn), then M transfers $(T_S)^n_{SF}$ to $(T_{bin})^n_{BF}$, and p_f, q_c, q_r from FF to SF and then to BF

   (b) If (fluids_burn), then M computes $(\boldsymbol{y})^n_{BF}$ from $(\boldsymbol{y})^n_{FN}$
   else (solids_burn), then M computes $(\boldsymbol{y})^n_{BF}$ from $(\boldsymbol{y})^n_{SN}$

2. Combustion code B: subcycles by repeating (a) through (c) — FA

   (a) Obtains $(p_f)^{n+\alpha}_{BF}$, $(q_c)^{n+\alpha}_{BF}$, $(q_r)^{n+\alpha}_{BF}$, $(T_{bin})^{n+\alpha}_{BF}$ (see 0a & 7g), $(\rho_s)^{n+\alpha}_{BF}$ (see 1a) , and$(\boldsymbol{y})^n_{BF}$ from M by extrapolation

   (b) Updates its solution to get $(\dot{r}_b)_{BF}$, $(S)_{BF}$, and $(T_{bout})_{BF}$

   (c) At the end of subcycling, sends $(\dot{r}_b)^{n+1}_{BF}$, $(S)^{n+1}_{BF}$, and $(T_{bout})^{n+1}_{BF}$ to M

3. Manager: (init_inbuff for S)

   (a) Transfers $(\boldsymbol{t}_s)_{FF}$ to $(\boldsymbol{t}_s)_{SF}$, $(q_r)_{FF}$ to $(q_r)_{SF}$, $(q_c)_{FF}$ to $(q_c)_{SF}$, $(C_{f,v})_{FF}$ to $(C_{f,v})_{SF}$, and $(T_s)_{SN}$ to $(T_s)_{SF}$ [The time index is n if iPredCorr==1 and is $n+1$ otherwise].

   (b) Transfers $(\dot{r}_b)^{n+1}_{BF}$ to $(\dot{r}_b)^{n+1}_{SF}$ and $(S)^{n+1}_{BF}$ to $(S)^{n+1}_{SF}$.

(c) Computes $(q_s)_{SF} = (q_r)_{SF} + (q_c)_{SF} + (\dot{m})_{SF}\,(T_s)_{SF}\left((C_s)_{SF} - (C_{f,v})_{SF}\right) + (S)_{SF}$ [The time index is n if iPredCorr==1 and is $n+1$ otherwise] [New for GEN3] .

4. Interface code invoked by M — FA, ALE

   Propagate undeformed boundary (on solid interface mesh) using Huygens' construction due to burning only. In this first implementation, the connectivity of the surface elements is assumed to remain unchanged (i.e., no interface nodes or elements are added or removed during the burning process). The reasoning is that Huygens' construction is valid for burning only, not for structural displacements, and that $\dot{r}_b$ is defined on the deformed configuration, whereas the solid code applies mesh motion to the undeformed configuration.

   (a) Assigns prop interface to be undeformed solid interface (or fluid interface if run in fluid-alone)

   (b) Computes normals $(\boldsymbol{N})_F^{n+1}$ of prop interface using $(\boldsymbol{x})_N^n$ (see 0b and 5d)

   (c) Propagates burning faces using mesh motion velocities $(\boldsymbol{v})_F^{n+1} = z\,(\dot{r}_b)_F^{n+1}\,(\boldsymbol{N})_F^{n+1}$ (see 3b & 5d), where $\boldsymbol{v}$ is $\bar{\boldsymbol{v}}$ for solid and $\boldsymbol{v}_m$ for fluid — Z

   (d) Transfers mesh motion from faces to nodes by geometric construction

   (e) Interpolates mesh motion to nonburning nodes

   (f) Sends $(\boldsymbol{v})_N^{n+1}$ to owner of prop interface to be used as boundary condition for mesh motion in volume in 5c or 7f

5. Solid code: subcycles by repeating the steps (a) through (c) — SA

   (a) Obtains $(\boldsymbol{t}_s)_{SF}^{n+\alpha}$, $(q_s)_{SF}^{n+\alpha}$ (by extrapolation) and $(\bar{\boldsymbol{v}})_{SN}^{n+\alpha}$ (by interpolation, 4f) from M

   (b) Solves for mesh motion, i.e., computes $\bar{\boldsymbol{v}}^{n+\alpha}$ and then $\bar{\boldsymbol{u}}^{n+\alpha}$ over the volume

   (c) Solves for structural motion in the solid, i.e., computes $\hat{\boldsymbol{v}}$, $\hat{\boldsymbol{u}}$ $\hat{\boldsymbol{a}}$, and T_s, using $(\boldsymbol{t}_s)_{SF}^{n+\alpha}$ and $(q_s)_{SF}^{n+\alpha}$

   (d) At the end of subcycling, computes $(\boldsymbol{v}_s)_{SN}^{n+1}$, and sends $(\rho_s)_{SF}^{n+1}$, $(\hat{\boldsymbol{u}})_{SN}^{n+1}$, $(\boldsymbol{u})_{SN}^{n+1}$ (total displacement), $(\boldsymbol{v}_s)_{SN}^{n+1}$, $(T_s)_{SN}^{n+1}$, and $(\boldsymbol{x})_{SN}^{n+1}$ to M

6. Manager: (post_update (a–b) for S and init_inbuff (c–e) for F)

   (a) If solid-ALE is on and z=1, then computes $(\dot{m})_{SF}^{n+1} = (\rho_s \Delta V)_{SF} \,/\, (A\Delta t)_{SF}$, where $(A)_{SF}^{n+1}$ is computed on deformed configuration and $(\Delta V)_{SF}^{n+1}$ (stores in $(\dot{m})_{SF}^{n+1}$) on undeformed — Z, ALE

   (b) Computes $(\boldsymbol{y})_{SN}^{n+1} = (\boldsymbol{x})_{SN}^{n+1} + (\hat{\boldsymbol{u}})_{SN}^{n+1}$ [Note: Must be after step (a).]

   (c) Transfers $(\boldsymbol{u})_{SN}^{n+1}$ to $(\boldsymbol{u})_{FN}^{n+1}$ and $(\boldsymbol{v}_s)_{SN}^{n+1}$ to $(\boldsymbol{v}_s)_{FF}^{n+1}$

   (d) Transfers $(T_{bout})_{BF}^{n+1}$ to $(T_{fin})_{FF}^{n+1}$, and $(\dot{r}_b)_{BF}^{n+1}$ to $(\dot{r}_b)_{FF}^{n+1}$

   (e) Computes $(\Delta\boldsymbol{u})_{FN}^{n+1} = \left((\boldsymbol{y})_{FN}^{0} + (\boldsymbol{u})_{FN}^{n+1}\right) - (\boldsymbol{y})_{FN}^{n}$ if fully-coupled,
   or if fluid-alone, computes as $(\Delta\boldsymbol{u})_{FN}^{n+1} = \Delta t\,(\bar{\boldsymbol{v}})_{N}^{n+1}$ (see 4f) — FA

   (f) If solid-ALE is on and $z = 1$, transfers $(\dot{m})_{SF}^{n+1}$ to $(\dot{m})_{FF}^{n+1}$; — Z, ALE
   otherwise, computes $(\dot{m})_{FF}^{n+1}$ as $(\rho_s)_{FF}^{n+1} \cdot (\dot{r}_b)_{FF}^{n+1}$ — FA

7. Fluid code: subcycles by repeating the steps (a) through (d) — FA

   (a) Obtains $(T_{fin})_{FF}^{n+\alpha}$ (see 2c) from M by interpolation

(b) Computes $(\rho_f)_{FF}^{n+\alpha}$, which depends on $(T_{fin})_{FF}^{n+\alpha}$, and sends $(\rho_f)_{FF}^{n+\alpha}$ and $(\boldsymbol{n}_f)_{FF}^{n+\alpha}$ to M

(c) Obtains the nominal mass flux $(\dot{m}^z)_{FF}^{n+\alpha} = (\dot{m})_{FF}^{n+\alpha}$, where $(\dot{m})_{FF}^{n+\alpha}$ and $(\dot{r}_b)_{FF}^{n+\alpha}$ are obtained by interpolation and z is the zoom factor Z, ALE
[Note: $(\dot{m}^z)_{FF}^{n+\alpha}$ and $(\dot{m})_{FF}^{n+\alpha}$ use the same buffer from fluid's perspective.]

(d) Obtains injection velocity at fluid faces from M, computed as

$$(\rho_f \boldsymbol{v}_f)_{FF}^{n+\alpha} = (\rho_f)_{FF}^{n+\alpha} (\boldsymbol{v}_s)_{FF}^{n+\alpha} + \left((\rho_f)_{FF}^{n+\alpha} (\dot{r}_b^z)_{FF}^{n+\alpha} - (\dot{m}^z)_{FF}^{n+\alpha}\right) (\boldsymbol{n}_f)_{FF}^{n+\alpha}$$

if solid-ALE is on, or computed as

$$(\rho_f \boldsymbol{v}_f)_{FF}^{n+\alpha} = (\rho_f)_{FF}^{n+\alpha} (\boldsymbol{v}_s)_{FF}^{n+\alpha} - (\dot{m}^z)_{FF}^{n+\alpha} (\boldsymbol{n}_f)_{FF}^{n+\alpha},$$

otherwise, where $\dot{r}_b^z = z\dot{r}_b$, $(\dot{m}^z)_{FF}^{n+\alpha}$ is same as in (c), and $(\boldsymbol{v}_s)_{FF}^{n+\alpha}$ is obtained by interpolation Z, ALE

(e) Obtains $(\Delta \boldsymbol{u})_{FN}^{n+\alpha}$ from M (see 6e)

(f) Solves for mesh motion at fluid nodes and field variables at face centers

(g) At the end of subcycling, sends $(C_{f,v})_{FF}^{n+1}$, $(p_f)_{FF}^{n+1}$, $(q_c)_{FF}^{n+1}$, $(q_r)_{FF}^{n+1}$, $(\boldsymbol{t}_f)_{FF}^{n+1}$, $(T_{fout})_{FF}^{n+1}$, $(\boldsymbol{y})_{FF}^{n+1}$, and $(\text{bflag})_{FF}^{n+1}$ to M

8. Manager: (post_update (a) for F and convergence check (b))

(a) If solid-ALE is on, computes $(\boldsymbol{t}_s)_{FF} = (\boldsymbol{t}_f)_{FF} - (\dot{m})_{FF} \left((\dot{m})_{FF} / (p_f)_{FF} - (\dot{r}_b)_{FF}\right) (\boldsymbol{n}_f)_{FF}$; ALE
otherwise, computes $(\boldsymbol{t}_s)_{FF} = (\boldsymbol{t}_f)_{FF} - (\dot{m})_{FF}^2 / (p_f)_{FF} (\boldsymbol{n}_f)_{FF}$ (see 0a & 7g) (conservation of linear momentum)

(b) With PC iterations, checks convergence on $\dot{m}$, $\boldsymbol{t}_s$, $\Delta \boldsymbol{u}$, and $\boldsymbol{v_s}$, all on fluid faces

9. If converged,

(a) Backs up $(\rho_s)_{BF}^n$, $(p_f)_{BF}^n$, $(q_c)_{BF}^n$, $(q_r)_{BF}^n$, $(T_{bin})_{BF}^n$, $(\boldsymbol{t}_s)_{SF}^n$, $(\bar{\boldsymbol{v}})_{SN}^n$, $(q_s)_{SF}^n$, $(r_b)_{FF}^n$, $(\dot{m})_{FF}^n$, $(\Delta u)_{FN}^n$, $(\boldsymbol{v}_s)_{FF}^n$, and $(T_{fin})_{FF}^n$

(b) Advances time stamp by $z\Delta t$, where z is zoom factor and Δt is system time step FA, SA

[Remark: With PC iterations, the manager always uses interpolation for obtaining values at $n + \alpha$ if iPred-Corr>1 (i.e. after the first PC iterations). At time 0, the manager sets values at $n + \alpha$ to the most recent values (i.e. uses constant extrapolation). This avoids requiring an initial guess for all quantities at time 0.]

4 Implementation

Each physical component (denoted by X) needs to provide and register with Roccom the following subroutines (these names are the strings registered to Roccom):

- initialize(G, initialTime, communicator, manInitHandle, win_surf, win_vol, obtainHandle): Allocate and initialize data structures internal to the component. Some modules may have additional parameters. This subroutine should also create two windows: one for the interface data, and the other for the internal (volume) data that need to be stored for restart and for predictor-corrector iterations. It is important for the module to initialize all outgoing data correctly (including nodal coordinates) by the end of this subroutine. During the initialization, the module should set bounds for the attribute they register.

- update_solution(G, currentTime, timeStep, handles): Update the solution for the time interval timeStep and subcycle if necessary. This subroutine must call a function update_inbuff_X provided by Rocman (described later) before each subcycle. At the end of this subroutine, it should update the outgoing interface buffers.

- compute_integrals(G, integrals): Compute the integrals (including volume, total mass, surface area, and so on) for sanity checking. This routine should be provided by fluids and solids codes only. The argument integrals is an array, whose content is defined in Rocman/include/rocmanf90.h.

- finalize(G): Deallocate memory and delete the windows.

The first argument G of these routines is an application-specific data object storing the "global data" of the module. Note that the store_state and restore_state are no longer necessary because they will be provided by Rocman instead. Each module should provide a subroutine RocX_load_module for creating a window containing these subroutines. The window name is the only argument of RocX_load_module.

NOTE: Change to Rocburn: do not pass the function handles from Rocman; pass them in Rocburn.

4.1 Manager

For each component X, Rocman provides the following callback routines (note that here G is the global variable of Rocman and is passed implicitly by *Roccom*):

- initialize(G, surfWinName, volWinName, ...): Create buffer space for the intermediate data described in Section 2.4. This routine takes one more input integer argument for fluids indicating whether the fluids code solves the Euler or Navior-Stokes equation. It also takes one more input integer argument for combustion indicating whether the combustion code expects heat-fluxes from fluids. When the fluids code solves Euler equation, the combustion code should not expect heat-fluxes from fluids. The handle of this routine is passed to the initialization routine of the component X.

- update_inbuff_bc(G, alpha [, level]): Fill in incoming buffers for boundary conditions of X by interpolation/extrapolation in time. The handle of this routine is passed to the update_solution routine of the component X, and it should be called inside update_solution before each subcycle. The third argument level is present only for fluids, and the subroutine updates mdot_alp and Tflm_alp when level==1 and updates rhofvf_alp otherwise.

- update_inbuff_gm(G, delta_alpha): Fill in the incoming buffers for grid motion of X by interpolation in time. In GEN3, this subroutine is only needed for fluids, and its handle is passed to the update_solution routine of fluids code. It should be called in update_solution before each subcycle.

For example, in the control flow, Step 2a–2d is update_solution of the Burn module, Step 2a is update_inbuff_Burn_bc.

4.2 Fluids

Fluids code must create a window that contains the following data attributes.

Name	Intention	Meaning	Type	Unit
'pf'	OUT	$(p_f)_{FF}^{n+1}$	DBL,DIM(:)	'Pa'
'qc'	OUT	$(q_c)_{FF}^{n+1}$	DBL,DIM(:)	'W/m^2'
'qr'	OUT	$(q_r)_{FF}^{n+1}$	DBL,DIM(:)	'W/m^2'
'Cfv'	OUT	$(C_{f,v})_{FF}^{n+1}$	DBL,DIM(:)	'J/(KgK)'
'rhof_alp'	OUT	$(\rho_f)_{FF}^{n+\alpha}$	DBL,DIM(:)	'kg/m^3'
'nf_alp'	OUT	$(\boldsymbol{n}_f)_{FF}^{n+\alpha}$	DBL,DIM(3,:)	N/A
'tf'	OUT	$(\boldsymbol{t}_f)_{FF}^{n+1}$	DBL,DIM(3,:)	'Pa'
'Tfout'	OUT	$(T_{fout})_{FF}^{n+1}$	DBL,DIM(:)	'K'
'bflag'	INOUT	$(\text{bflag})_{FF}^{n+1}$	INT,DIM(:)	N/A
'bcflag'	OUT	BC type	INTEGER	N/A
'du_alp'	IN	$(\Delta u)_{FF}^{n+\alpha}$	DBL,DIM(3,:)	'm'
'mdot_alp'	IN	$(\dot{m})_{FF}^{n+\alpha}$	DBL,DIM(:)	'kg/(m^2s)'
'rhofvf_alp'	IN	$(\rho_f \boldsymbol{v}_f)_{FF}^{n+\alpha}$	DBL,DIM(3,:)	'kg/(m^2s)'
'Tfin_alp'	IN	$(T_{fin})_{FF}^{n+\alpha}$	DBL,DIM(:)	'K'

The attributes du_alp should be allocated for all panes; rhof_alp, nf_alp, tf, mdot_alp, and rhofvf_alp should be allocated for all interacting panes; pf, qr, qc, and Tf should be allocated for burning panes. Note that the fluids code need to set correct values for $(\rho_f)_{FF}^{n+\alpha}$ and $(\boldsymbol{n}_f)_{FF}^{n+\alpha}$ before calling update_inbuff_bc_fluid. A pane of the fluid interface window is either ignitable (bcflag=1), non-ignitable (bcflag=0), noninteracting (bcflag=2).

4.3 Combustion

The combustion module inherits the mesh and data attributes of the ignitable panes of the fluid interface. Furthermore, it needs to create the following additional attributes.

Name	Intention	Meaning	Type	Unit
'rb'	OUT	$(r_b)_{BF}^{n+1}$	DBL,DIM(:)	'm/s'
'Tbout'	OUT	$(T_{bout})_{BF}^{n+1}$	DBL,DIM(:)	'K'
'S'	OUT	$(S)_{BF}^{n+1}$	DBL,DIM(:)	'W/m^2'
'bflag'	INOUT	$(\text{bflag})_{BF}^{n+1}$	INT,DIM(:)	N/A
'pf_alp'	IN	$(p_f)_{BF}^{n+\alpha}$	DBL,DIM(:)	'Pa'
'qc_alp'	IN	$(q_c)_{BF}^{n+\alpha}$	DBL,DIM(:)	'W/m^2'
'qr_alp'	IN	$(q_r)_{BF}^{n+\alpha}$	DBL,DIM(:)	'W/m^2'
'rhos_alp'	IN	$(\rho_s)_{BF}^{n+\alpha}$	DBL,DIM(:)	'kg/m^3'
'Tbin_alp'	IN	$(T_{bin})_{BF}^{n+\alpha}$	DBL,DIM(:)	'K'
'centsrs'	IN	$(\boldsymbol{y})_{BF}^{n+\alpha}$	DBL,DIM(3,:)	'm'

4.4 Solids

The solid code needs to create the following attributes in an interface window.

Name	Intention	Meaning	Type	Unit
'rhos'	OUT	$(\rho_s)_{SF}^{n+1}$	DBL,DIM(:)	'kg/m^3'
'Cs'	OUT	$(C_s)_{SF}^{n+1}$	DBL,DIM(:)	'J/(KgK)'
'Ts'	OUT	$(T_s)_{SN}^{n+1}$	DBL,DIM(:)	'K'
'u'	OUT	$(\boldsymbol{u})_{SN}^{n+1}$	DBL,DIM(3,:)	'm'
'vs'	OUT	$(\boldsymbol{v}_s)_{SN}^{n+1}$	DBL,DIM(3,:)	'm/s'
'uhat'	OUT	$(\hat{\boldsymbol{u}})_{SN}^{n+1}$	DBL,DIM(3,:)	'm'
'bcflag'	OUT	BC type	INTEGER	N/A
'ts_alp'	IN	$(\boldsymbol{t}_s)_{SN}^{n+\alpha}$	DBL,DIM(1or3,:)[1]	'Pa'
'vbar_alp'	IN	$(\bar{\boldsymbol{v}})_{SN}^{n+\alpha}$	DBL,DIM(3,:)	'm/s'
'qs_alp'	IN	$(q_s)_{SF}^{n}$	DBL,DIM(:)	'W/m^2'

The interface code moves the entire *undeformed* solid surface mesh (including the part not on fluid-solid interface). Solids must create a window for the entire surface, and the nodal coordinates of the window should contain values for undeformed configurations. The deformed configuration can be obtained by Rocman by adding $\hat{\boldsymbol{u}}$ onto nodal coordinates. A pane of the fluid interface window is either ignitable (bcflag=1), non-ignitable (bcflag=0), noninteracting (bcflag=2).