The following pseudo-code blocks summarize a typical *MIRGE-Com* simulation application for a viscous, reactive gas mixture.

1 Simulation Infrastructure

Constructs provided (mostly) by *MIRGE-Com*.

1.1 Simulation Driver

The simulation driver construct is where the simulation execution begins and ends. Is is the so-called *main* routine from which all others are called. The simulation driver is typically written wholly by the domain user using pre-built pieces provided by the *MIRGE-Com* library.

For simplicity of illustration, the driver control-flow is presented here as a library-provided construct, where the user-written pieces further customize the simulation. Those user-written pieces are shown here in bold. Two abstract user-written constructs executed as part of the driver are:

- **User_Init** - abstract construct that initializes the simulation from scratch, or from restart files, setting up the initial condition, boundary conditions, and current simulation epoch.
- **User_Finalize** - abstract construct that finalizes the simulation, performing, for example, a save of the final state.

The remaining user-written functions are further explained in dedicated sections to follow.

Algorithm 1 Simulation Driver

$CV_0, DV_0, TV_0, n, t, t_{final}, dt \leftarrow$ **User_Init**()
$Tseed_0 \leftarrow DV_0.temperature$
$S_0 \leftarrow [CV_0, Tseed_0]$
$S_n \leftarrow$ STEPPER(t, t_{final}, n, dt, S_0, **User_RHS**, **User_PreStep**, **User_PostStep**)
$[CV_n, Tseed_n] \leftarrow S_n$
$DV_n \leftarrow$ EOS($CV_n, Tseed_n$)
$TV_n \leftarrow$ TRANSPORT(CV_n, DV_n)
User_Finalize() ▷ save final state

The general simulation state is represented by S, and the domain-specific simulation quantities are as follows:

- CV - vector of fluid conserved quantities $(\rho, \rho\vec{V}, \rho E, \rho Y_\alpha)$
- DV - vector of fluid state-dependent quantities, e.g., *pressure*, *temperature*, *sound speed*
- TV - vector of transport properties, e.g., *viscosity*, *species diffusivities*

- Tseed - a mechanism for propagating the fluid temperature from the last step in order to *seed* subsequent temperature calculations.

Two domain-specific constructs that are not detailed here are the *EOS* and *Transport* constructs. These are fluid model-specific constructs that are also provided by the user.

1.2 Simulation Stepper

This library-provided routine marches the simulation state S forward in time using the user's chosen time integration method, and user-provided pre-and-post-step utilities, and RHS.

Algorithm 2 Stepper

procedure STEPPER($t, t_{final}, dt, n, S_n,$ **RHS**, **PreStep**, **PostStep**)
  while $t < t_{final}$ **do**
    $S_n, dt \leftarrow$ PRESTEP(n, t, dt, S_n)
    $S_{n+1} \leftarrow$ TIMEINTEGRATOR($t, dt, S_n,$ **RHS**)
    $t \leftarrow t + dt$
    $n \leftarrow n + 1$
    $S_{n+1}, dt \leftarrow$ POSTSTEP(n, t, dt, S_{n+1})
    $S_n \leftarrow S_{n+1}$
  end while
end procedure

1.3 Time Integrators

A collection of time integrators are provided by *MIRGE-Com*.

Algorithm 3 RK4 Time Integrator

procedure TIMEINTEGRATOR($t, dt, S,$ **RHS**)
  $k1 \leftarrow \mathrm{RHS}(t, S)$
  $k2 \leftarrow \mathrm{RHS}(t + \frac{dt}{2}, S + \frac{dt}{2}k1)$
  $k3 \leftarrow \mathrm{RHS}(t + \frac{dt}{2}, S + \frac{dt}{2}k2)$
  $k4 \leftarrow \mathrm{RHS}(t, S + dt\ k3)$
  return $S + dt\frac{(k1+2k2+2k3+k4)}{6}$
end procedure

2 User/Domain Functions

The user/domain functions are those that customize the simulation to the user's specific case, and, in-general, are the following functions:

- **User_PreStep** - Proper function passed to and called by the library-provided *Stepper* before a time integration step is performed.
- **User_RHS** - Proper function passed to and called by the library-probided *TimeIntegrator*. The **User_RHS** function provides the time-rate-of-change for the conserved quantities used by the *TimeIntegrator* to advance the state forward in time.
- **User_PostStep** - Proper function passed to and called by the library-provided *Stepper* after a time integration step is completed, and before the next time integration step.

2.1 RHS

Algorithm 4 User's RHS Function

procedure RHS(t, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $\Psi \leftarrow [CV, DV, TV]$ ▷ forms fluid state Ψ
  $[\Psi_q] \leftarrow \textsc{Project}(\Psi)$ ▷ project Ψ to quadrature/boundaries
  return $[\Sigma\textsc{Op}(t, [\Psi_q]) + \Sigma\textsc{Sources}(t, [\Psi_q]), 0]$ ▷ Note Tseed RHS = 0
end procedure

The function(s) *Op* may include the compressible Navier-Stokes operator, artificial viscosity, etc. *Sources* would include production rates for reactant and product mixture species, and possibly others.

2.2 Prestep and Poststep Callbacks

The callbacks are user-provided functions where things such as I/O, simulation health checking, and timestep computations are performed. For

Algorithm 5 User's Prestep Callback

procedure PreStep(n, t, dt, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $\Psi \leftarrow [CV, DV, TV]$ ▷ forms fluid state Ψ
  $dt \leftarrow \textsc{SimTimestep}(t, t_{final}, dt, \Psi)$
(...) ▷ I/O, Health, etc
  return $[S, dt]$
end procedure

Algorithm 6 User's Poststep Callback

procedure PostStep(n, t, dt, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $Tseed \leftarrow DV.temperature$
  $S \leftarrow [CV, Tseed]$ ▷ Updates temperature seed
  return $[S, dt]$
end procedure

3 Species Limited Versions

In the current version of the algorithms with species mass-fraction-limited, no change to the library-provided infrastructure are required. The current *LimitSpecies* function restricts the species mass fractions to $[0, 1]$ and calculates a source term designed to help drag the running fluid state back to a state such that the species mass fractions $Y_\alpha \in [0, 1]$. The changes to support this limiting are restricted to the main user-provided constructs, which are modified as follows:

Algorithm 7 User's RHS Function w/Species Limiting

procedure RHS(t, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $CV, L_s \leftarrow \textsc{LimitSpecies}(CV, DV)$ ▷ gets limited CV and source
  $DV \leftarrow \mathrm{EOS}(CV, DV.temperature)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $\Psi \leftarrow [CV, DV, TV]$ ▷ forms fluid state Ψ
  $[\Psi_q] \leftarrow \textsc{Project}(\Psi)$ ▷ project Ψ to quadrature/boundaries
  return $[\Sigma\textsc{Op}(t, [\Psi_q]) + \Sigma\textsc{Sources}(t, [\Psi_q]) + L_s, 0]$
end procedure

Algorithm 8 User's Prestep Callback

procedure PRESTEP(n, t, dt, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $CV, L_s \leftarrow \textsc{LimitSpecies}(CV, DV)$
  $DV \leftarrow \mathrm{EOS}(CV, DV.temperature)$
  $TV \leftarrow \textsc{Transport}(CV, DV)$
  $\Psi \leftarrow [CV, DV, TV]$ ▷ forms fluid state Ψ
  $dt \leftarrow \textsc{SimTimestep}(t, t_{final}, dt, \Psi)$
(...) ▷ I/O, Health, etc
  return $[S, dt]$
end procedure

Algorithm 9 User's Poststep Callback

procedure POSTSTEP(n, t, dt, S)
  $[CV, Tseed] \leftarrow S$
  $DV \leftarrow \mathrm{EOS}(CV, Tseed)$
  $CV, L_s \leftarrow \textsc{LimitSpecies}(CV, DV)$
  $DV \leftarrow \mathrm{EOS}(CV, DV.temperature)$
  $Tseed \leftarrow DV.temperature$
  $S \leftarrow [CV, Tseed]$ ▷ Updates temperature seed
  return $[S, dt]$
end procedure