Rocburn_PY Developer Guide

September 18, 2015

License

Contents

1 Introduction **2**

2 Combustion and Ignition at the Propellant Surface **2**

3 Review of Thermal Transient Theory Inside *Rocburn_PY* **4**

3.1 Regression Speed 4

3.2 Heat Flux to the Surface 5

3.2.1 Pre-Ignition Phase Along Propellant Surfaces 5

3.2.2 Post-Ignition Phase Along Propellant Surfaces 6

3.2.3 Non-Propellant Surfaces 7

4 Numerical Solution of Unsteady Heat Conduction **8**

5 Physics Implementation in *Rocburn_PY* Source **9**

5.1 Problem Setup 10

5.2 Parameter Initialization at $t = 0$ 11

5.3 Parameter Update at Time t 13

6 Input Control File and Using *Rocburn_PY* **15**

6.1 Summary of Input Parameters 16

6.1.1 Combustion Model Parameters 17

6.1.2 Ignition Model Parameters 17

6.1.3 Divergence Boundaries 18

6.1.4 Grid Parameters 19

6.1.5 Lookup Table Parameters 19

6.2 Working *Rocburn_PY* Input File 20

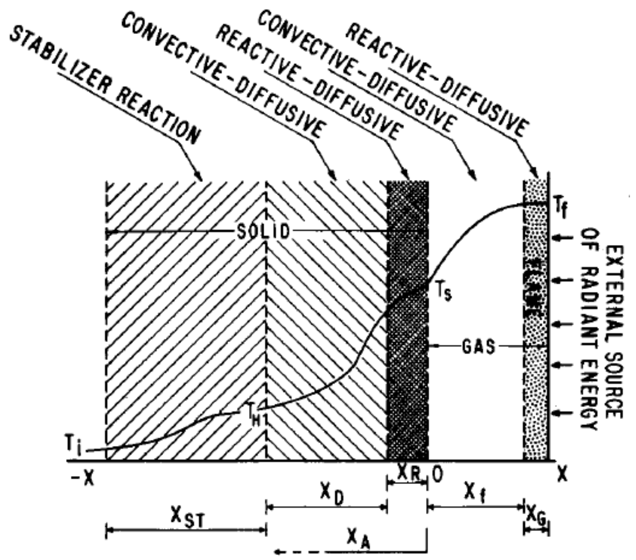
1 Introduction

Rocburn_PY is defined by one-dimensional physics, appropriate for a homogeneous propellant, and it is assumed that its parameters can be chosen to represent the heterogeneous case in some fashion. The physics that is not accounted for in any plausible way in this strategy includes the non-planar surface regression and the diffusion flames in the combustion field. *Rocburn_PY* is a numerical module inside *Rocstar* that simulates the thermal transient effects in the wall-normal direction of a homogeneous medium found in solid propellant rocket motors (e.g., insulated walls, propellant, nozzle, etc.). The time-dependent temperature in the solid is computed using a one-dimensional model that takes into account a temperature gradient normal to (only) the burning surface. Thus, the solid is essentially treated as a semi-infinite solid, and the unsteady heat equation is solved in the wall-normal direction. The *Rocburn_PY* table feature is created by averaging the solid phase heat conduction over a plane that is parallel to the mean surface, accounting for rotational terms that arise due to the uneven surface. Boundary conditions are applied at the propellant surface and include, most importantly, the heat flux from the combustion field. This heat flux is not modeled but is contained in a universal lookup generated by another module in *Rocstar*. Information about combustion and ignition of solid propellants is available elsewhere [Ibiricu and Williams (1975); Blomshield et al. (1997); Ward et al. (1998); Brewster et al. (2000); Chen et al. (2002); Jackson and Buckmaster (2002); Massa et al. (2002, 2004, 2005)].

2 Combustion and Ignition at the Propellant Surface

Before presenting the theory behind the *Rocburn_PY* implementation, this section discusses the ideas of propellant combustion and ignition more broadly. The thermal transient theory inside *Rocburn_PY* is abstract, so presentation of this broader picture will help to concretize the physical phenomena taking place in the different regions, each modeled in its own way. The following figure and associated description are taken directly from [Ibiricu and Williams (1975)].

> The combustion model is illustrated schematically in Figure 2.1. The solid phase occupies the region $x < 0$ and the gas phase the region $x > 0$. Externally provided radiation is incident on the propellant from the gas. The temperature profile has been drawn for the unperturbed state of zero radiant flux. Allowance is made for gaseous and solid reactions, each being presumed to have activation energies sufficiently high for reactive–diffusive and convective–diffusive zones to be identifiable [Williams (1973)]. In addition, a stabilizer reaction zone is included for propellants containing stabilizers that react exothermically at low temperatures [Huggett et al. (1960); Kovalskii et al. (1967)]. Regression rates are assumed to be controlled by the gaseous or solid reaction, or by a combination thereof, but not by the stabilizer reaction, whose effect is treated purely energetically.
>
> Although it may not be apparent, the model encompasses conventional views of double-base propellant combustion [Huggett et al. (1960)]. The convective–



	Characteristic Lengths
Stabilizer Heat Release	$X_{ST} \geq X_D$
Solid Heat Conduction	$X_D = \lambda / r \rho c$
Solid Gasification	$X_R = X_D R T_S / E_S$
Gas Heat Conduction	$X_f = \lambda_g / r \rho c_p$
Gas Reaction	$X_G = X_f R T_f / E$
Radiation Absorption	$X_A = 1/\mu$

Figure 2.1: Schematic illustration of combustion model from [Ibiricu and Williams (1975)]. Variable descriptions: solid thermal conductivity, λ; burn rate, r; density, ρ; solid heat capacity per unit mass, c; universal gas constant, R; temperature of solid surface, T_s; activation energy for solid phase reaction, E_s; gas thermal conductivity, λ_g; gas specific heat at constant pressure, c_p; flame temperature, T_f; activation energy for gas phase reaction, E; and absorption coefficient for solid, μ.

diffusive zone in the gas may represent the "dark zone," and the reactive–diffusive zone in the solid is a combination of the subsurface and "fizz" reaction zones of the classical description. Placement of the surface on the hot side of this last zone, and calling it the "solid reaction zone" are merely conventions ... Experimental demarkation of a boundary between subsurface and fizz zones is inconclusive, and two-phase flow analyses needed in attempting to resolve the issue are very difficult. If the demarkation can be achieved, then a boundary temperature can be identified which conventionally has been called the "surface temperature" and which lies below the T_s of Figure 2.1 when the reactive–diffusive zone of the solid encompasses the fizz zone. Since rates of fizz-zone reactions may be expected to depend on pressure, there should be no hesitance to assign pressure dependence to rates in the present "solid reaction zone." The model is capable of retaining or excluding the break in the temperature curve shown at $x = 0$. [Ibiricu and Williams (1975)]

3 Review of Thermal Transient Theory Inside *Rocburn_PY*

The equation that governs the unsteady heat conduction in a homogeneous medium with a surface regressing normal to itself is given by

$$\rho_c c_p \left(\frac{\partial T}{\partial t} + r_b \frac{\partial T}{\partial x} \right) = \lambda_c \frac{\partial^2 T}{\partial x^2}. \tag{3.1}$$

Here x is the distance normal from the surface with range $-x_{\text{end}} < x < 0$, r_b is the speed of the regressing surface, T is the temperature, and t is the time. The material properties λ_c, ρ_c, and c_p are the thermal conductivity, density, and specific heat of the solid, respectively.

The appropriate boundary condition at $-x_{\text{end}}$ then becomes

$$T(-x_{\text{end}}) = T_\infty. \tag{3.2}$$

where T_∞ is the supply (or ambient) temperature. The boundary condition at $x = 0$ may depend on if the expression is evaluated from the solid phase side (0^-) or the gas phase side (0^+) of the surface. For the convective–diffusion zone at the surface,

$$T(0^-) = T(0^+) \equiv T_s. \tag{3.3}$$

where T_s is the surface temperature. The derivative with respect to x, evaluated from the solid phase side (0^-), is given by

$$\frac{\partial T}{\partial x}(0^-) = g(T_s, T_g) \tag{3.4}$$

The flux function $g(T_s, T_g)$ is the heat flux from the gas phase to the solid phase surface where T_g is the gas phase temperature. As might be implied, the heat flux depends on the surface temperature of the solid as well as the gas phase “fluid” lying above the surface. The actual form of the flux g depends on whether the solid is a propellant, a no-slip wall, or an ablating wall. If the solid is a propellant, the form of g also differs for times before and after ignition.

3.1 Regression Speed

For a moving surface the regression speed r_b can be specified by one of two forms, as given in eqs (3.5) and (3.6), respectively. Trivially r_b is zero if the surface is not regressing. A simple expression for r_b can be defined by an experimentally determined power law for burning,

$$r_b = A_p \left(\frac{P}{P_{\text{ref}}} \right)^{n_p} \tag{3.5}$$

where A_p is the steady state regression speed at reference pressure P_{ref}, n_p is the pressure exponent, and P is the input pressure from the chamber, and A_p and n_p are empirical fits to data. Alternatively r_b can be defined by a one-step Arrhenius pyrolysis model,

$$r_b = A_c \exp\left[-\frac{E_c}{R_u T_s}\right] \tag{3.6}$$

where A_c is the exponential prefactor, E_c is the pyrolysis activation energy for the solid (condensed) phase, R_u is the universal gas constant, and T_s is the surface temperature.

3.2 Heat Flux to the Surface

As discussed above, the actual form of the flux $g(T_s, T_g)$ depends on whether the solid is a propellant, a no-slip wall, or an ablating wall. If the solid is a propellant, the form of g also differs for times before and after ignition.

3.2.1 Pre-Ignition Phase Along Propellant Surfaces

If the temperature is lower than the ignition temperature, then no mass flux is injected at the propellant surface, and a turbulent boundary layer separates the inviscid field from the propellant surface. Because of this situation, it is assumed that the turbulent boundary layer along the propellant surface is not resolved numerically, and the flux to the (non-ignited) propellant surface is approximated by integrating the Euler equations down to the surface. In such case $g(\cdots)$ is calculated as

$$g(T_s, T_g) = \frac{h(T_g - T_s)}{\lambda_c}. \tag{3.7}$$

The film coefficient, h, is expressed in terms of the Stanton number St,

$$h = (U_\infty \rho_g c_p)\mathsf{St} \tag{3.8}$$

where U_∞ is the free-stream velocity and ρ_g is the density of the gas. For the boundary layer with constant free-stream velocity and wall temperature, the Stanton number is given by the Prandtl number Pr and the Reynolds number Re:

$$\mathsf{St} = 0.0287(\mathsf{Pr})^{-2/5}(\mathsf{Re}_x)^{-1/5}. \tag{3.9}$$

Note that here, the Reynolds number is based on the streamwise coordinate x. Expanding these definitions gives the following expression for the film coefficient.

$$h = 0.0287 \rho_g U_\infty c_p \left(\frac{x \rho_g U_\infty \mu_g c_p^2}{\lambda_g^2}\right)^{-1/5} \tag{3.10}$$

where λ_g is the thermal conductivity of the gas and μ_g is the viscosity. Such a form is assumed valid for the general case in which both the wall temperature and free-stream velocity vary in x and in time, using the numerically evaluated outer edge solution as the free-stream temperature and velocity. Finally, note that

$$\frac{\partial g}{\partial T_s} = -\frac{h}{\lambda_c} \tag{3.11}$$

where λ_c is the thermal conductivity of the solid as before.

3.2.2 Post-Ignition Phase Along Propellant Surfaces

When the surface temperature reaches the ignition value, the surface cell is switched from a non-ignition surface to a burning cell,[a] and the heat flux to the surface is evaluated by integrating the steady state convection–diffusion–reaction equations for the gas phase. Activation energy asymptotic theory is used to relate the flame temperature $T_\star$ to the surface temperature.

$$\frac{1}{2}\frac{E_g}{R_u}\left[\frac{1}{T_\star^0} - \frac{1}{T_\star}\right] = \frac{E_c}{R_u}\left[\frac{1}{T_s^0} - \frac{1}{T_s}\right] \tag{3.12}$$

where the superscript zero (i^0) indicates the steady state values and the subscript star ($i_\star$) are the values at the outflow plane. Therefore $T_\star^0$ is the steady flame temperature, and the steady state surface temperature T_s^0 is taken by comparing the pyrolysis law in eq (3.6) to the experimental burn power law in eq (3.5). As before with eq (3.6), E_c/R_u is the scaled activation energy for solid phase, and likewise E_g/R_u is the scaled activation energy for gas phase (i.e., temperatures).

Once the relationship between flame temperature and surface temperature has been established, the integration of the convection–diffusion–reaction equations, together with the connection boundary condition at the interface, yields the following expression for $g(\cdots)$.

$$g(T_s, T_g) = \frac{r_b}{\alpha_c}\left[T_\star^0 - T_\infty + T_s - T_\star\right] \tag{3.13}$$

where α_c is the thermal diffusivity of the solid: $\alpha_c = \lambda_c/(\rho_c c_p)$. (See eq (3.1).) A result of activation energy asymptotic theory is that the injection temperature is equal to the flame temperature: $T_{\text{inject}} = T_\star$. Finally, note that

$$\frac{\partial g}{\partial T_s} = \frac{E_c}{R_u}\frac{1}{T_s^2}g + \frac{r_b}{\alpha_c}\left[1 - 2\frac{E_c}{E_g}\frac{T_\star^2}{T_s^2}\right]. \tag{3.14}$$

Recent improvements to *Rocburn_PY* now allow for a lookup table to compute the flux g rather than using the descriptions given above. Note that the lookup table is *propellant*

[a]That is, it is assumed that the transition events from ignition to complete combustion occur on a time scale that is too small to resolve numerically, and so the ignition event is treated as a on/off switch.

specific and requires significantly more work. For this reason the approach discussed above is sufficient for most purposes. Enabling the lookup table function in the *Rocburn_PY* input file (`TABUSE = 1`) is *strongly* discouraged.

3.2.3 Non-Propellant Surfaces

For the case of a no-slip wall or an ablating surface, the assumption is that the turbulent boundary layer will be resolved with enough accuracy as to capture the local heat flux to the wall from the gas phase. The solid phase equation is as before.

Rocflo solves the compressible equations using an explicit time integrator. In particular, for the cell adjacent to the surface, the cell-centered temperature is updated as

$$T_1^{n+1} = T_1^n + \Delta t \, \text{RungeKutta\_RHS}(T_s^n, T_1^n, T_2^n) \tag{3.15}$$

and will not be discussed in detail here.[b]

With T_1^{n+1} known from the gas phase, the flux function g can be written as

$$\begin{aligned} g &= \lambda_g \frac{\partial T}{\partial x} && (3.16) \\ &\approx \lambda_g^{n+1} \left(\frac{T_1^{n+1} - T_s^{n+1}}{\Delta x} \right), && (3.17) \end{aligned}$$

where Δx is the distance from T_1^{n+1} to the surface, i.e., it is just half the cell height in the wall-normal direction. The expression for the gas phase thermal conductivity at the surface can be expanded to

$$\lambda_g^{n+1} = \lambda_g^n + \left(\frac{\partial \lambda_g}{\partial T_s} \right)^n \left(T_s^{n+1} - T_s^n \right). \tag{3.18}$$

Thus, the flux connection condition in eq (3.4) at $x = 0$ becomes

$$\frac{\lambda_c}{2\Delta z} \left[-3T_s^{n+1} + 4T_{1,c}^{n+1} - T_{2,c}^{n+1} \right] \frac{dz}{dx} = \left[\lambda_g^n + \left(\frac{\partial \lambda_g}{\partial T_s} \right)^n \left(T_s^{n+1} - T_s^n \right) \right] \left(\frac{T_{1,g}^{n+1} - T_s^{n+1}}{\Delta x} \right), \tag{3.19}$$

and the subscripts c and g denote values in the solid (condensed) phase and the gas phase, respectively. Note that the flux function g is quadratic in T_s^{n+1}, so only the correct root need be selected.

[b] In actuality *Rocflo* solves the conservative form of the equations, not the primitive form. Eq (3.15) is only used here for illustration purposes.

4 Numerical Solution of Unsteady Heat Conduction

To solve for unsteady heat conduction in eq (3.1), subject to the boundary conditions in eqs (3.2) – (3.4), a computational grid is introduced with a cluster of points close to the surface $x = 0$. The code uses either an exponential grid or a boundary layer type grid. In either case, let $z = z(x)$ be the mapping. Then eq (3.1) transforms to

$$\rho_c c_p \left(\frac{\partial T}{\partial t} + r_b \frac{\partial T}{\partial z} \frac{dz}{dx} \right) = \lambda_c \left[\frac{\partial^2 T}{\partial z^2} \left(\frac{dz}{dx} \right)^2 + \frac{\partial T}{\partial z} \frac{d^2 z}{dx^2} \right]. \tag{4.1}$$

The flux boundary condition becomes

$$\frac{\partial T}{\partial z} \frac{dz}{dx} \left(0^- \right) = g \left(T_s, T_g \right). \tag{4.2}$$

The transformed unsteady heat equation for the interior points is solved numerically using a simple, first-order time integrator of the form

$$T^{n+1} = T^n \;+\; \Delta t \left[\alpha_c \left(\frac{\partial T}{\partial z} \frac{d^2 z}{dx^2} + \frac{\partial^2 T}{\partial z^2} \left(\frac{dz}{dx} \right)^2 \right) - r_b \frac{\partial T}{\partial z} \frac{dz}{dx} \right], \tag{4.3}$$

where $\alpha_c = \lambda_c/(\rho_c c_p)$ is the thermal diffusivity as before. A first-order time integrator is used because the timestep is typically small, and computational speed dictates that the integrator be fast. The spatial derivatives are computed using standard second-order central difference schemes. In all of test simulations, this approach was deemed sufficient.

The surface temperature at timestep $n + 1$, i.e., T_s^{n+1}, is found from the flux boundary condition. Note, however, that the equation is nonlinear in the surface temperature T_s. Since the timesteps are small, a full Newton method is not used; rather the g function is expanded about the previous timestep of T_s using Taylor's Theorem:

$$g(T_s^{n+1}, T_g^{n+1}) \approx g(T_s^n, T_g^n) + \left. \frac{\partial g}{\partial T_s} \right|_{T_s^n} \left(T_s^{n+1} - T_s^n \right). \tag{4.4}$$

Then using a standard one-sided, second-order finite difference scheme, the updated surface temperature is expressed as

$$T_s^{n+1} = \frac{4T_2^{n+1} - T_3^{n+1} - \frac{2\Delta z}{dz/dx} g + \frac{2\Delta z}{dz/dx} \frac{\partial g}{\partial T_s} T_s^n}{3 + \frac{2\Delta z}{dz/dx} \frac{\partial g}{\partial T_s}}. \tag{4.5}$$

The value of $g(T_s^n)$ and $\partial g/\partial T_s(T_s^n)$, which is computed in the subroutine `GFUN` (Section 5.3), depends on the specific conditions selected by the user; a more detailed discussion is presented in Section 6.

5 Physics Implementation in *Rocburn_PY* Source

Rocburn_PY is a module within the larger multiphysics code *Rocstar*. Its one-dimensional model is called a series of times, iterating over every surface point (or cell). The one-dimensional approach described above implies that each surface point is treated independently of the others. Specifically for the propellant surface, eq (3.1) is integrated for each point on the gas–solid interface.

The burn update is invoked by the main flow solver for each surface point at each timestep of the unsteady simulation. The temperature outside the turbulent boundary layer and the film coefficient are passed as input from the main solver to the burn update module, which computes the burn rate and the temperature of the injected gas used in the fluid phase boundary conditions. Porous surface boundary conditions are imposed for $T_s > T_{\text{ignition}}$ while solid surface boundary conditions are applied for $T_s < T_{\text{ignition}}$.

The unsteady heat conduction equation, eq (3.1), is discretized using second-order, finite differences in the spatial directions and a first-order, forward difference scheme in time. The very thin temperature boundary layer associated with the heat wave traveling into the solid requires a stretched grid to properly resolve the heat flux at the surface. To preserve second-order spatial accuracy, a computational mapping is used. The von Neumann boundary condition at the fluid–solid interface is imposed by solving a discrete form of eq (3.4). The first derivative is discretized using a standard second-order, one-sided formula and the nonlinear function g is expanded in a Taylor's series up to first order to obtain an explicit expression for the temperature at the surface, i.e., T_s.

For non-propellant surfaces, the treatment is similar to that just described except that the flux condition eq (3.4) is treated differently. In fact it can be shown that the numerical approximation leads to a quadratic function for the surface temperature, and the correct root must be taken. It is assumed, however, that, in the gas phase, the resolution is such that the heat flux can be adequately captured for time-accurate solutions.

As might be implied from above, the two main functions of *Rocburn_PY* are to pass information to the main solver upon initialization (e.g., temperature outside the turbulent boundary layer and the film coefficient) and then to update and compute the burn rate and heat flux as a function of time. The implementation is straightforward, and *Rocburn_PY* only has three main files and a limited number of subroutines. Information specific to the various files and subroutines are given in more detail below.

setup_py.f90 Reads user parameters from input file, sets up the (internal *Rocburn_PY*) grid and timestep information, and reads the data from the table (if enabled)

init_py.f90 Sets up the initial conditions for $t = 0$ and determines if any cells are set to burning upon initialization

update_py.f90 Updates the various temperatures and variables as functions of time, determines if new cells are burning (i.e., $T_s > T_{\text{ignition}}$), and calculates the flux g

5.1 Problem Setup

```
                              setup_py.f90

MODULE SETUP_PY

  USE data_py

CONTAINS

  SUBROUTINE get_time_step_1d(bp,rb,Toa,dt_max)
  SUBROUTINE burn_init_0d(bp,comm,IN_DIR,nx_read,To_read)
  SUBROUTINE read_properties(bp,IN_DIR)
  SUBROUTINE grid(bp)
  SUBROUTINE READTABLE(bp,IN_DIR)
  SUBROUTINE STEADYTEMP(bp,TIN,pIN,out,rb,g)
  SUBROUTINE CHECK_INPUT_RANGE(bp)
  SUBROUTINE burn_finalize_0d(bp)
```

`get_time_step_1d(bp,rb,Toa,dt_max)`

- Variable `dt_max = delt_max`

`burn_init_0d(bp,comm,IN_DIR,nx_read,To_read)`

- Call `read_properties` to read in the propellant thermophysical properties
- Set the return values `nx_read = numx` and `To_read = To` that are read from input
- Call `grid` to set up the computational grid
- If a table is to be read from input, then call `readtable`

`read_properties(bp,IN_DIR)`

- Read in all 29 required input parameters from RocburnPYControl.txt and write them to the screen for verification
- Note that the 29 parameters must be in the proper order and that the input file is named appropriately
- Input parameters are discussed in detail in Section 6
- Call `CHECK_INPUT_RANGE` to verify input parameters

`grid(bp)`

- Variables `gridtype = igrid`, `numx = numx`, `beta = beta`, `xmax = xmax`, and `delz = 1 / (numx-1)`, which are all read from the input file

- Set up x- and z-coordinate arrays
- Set up first and second derivatives of z, i.e., zx and zxx, respectively
- Create either exponential or boundary layer grid, depending on `igrid` input parameter choice
- Calculate maximum Δt for time integration
- Call `STEADYTEMP` to get appropriate state state temperature information

`READTABLE(bp,IN_DIR)`

- If the table read option is enabled in the input file, read the name of the file with the table information, which is also specified in the input file
- Allocate table fields and work fields

`STEADYTEMP(bp,TIN,pIN,out,rb,g)`

- Variable `To = To`, which is read from the input file
- Call `polin2` (several times) to pass in information read from the table
- Variable `out = g - (Tin - To)*rb / alfa`

`CHECK_INPUT_RANGE(bp)`

- Check that 14 selected parameters of the total 29 lie within the proper range
- Input parameters and their associated ranges are discussed in detail in Section 6

`burn_finalize_0d(bp)`

- Deallocate `bp` global array that contains data read from input

5.2 Parameter Initialization at $t = 0$

```
                              init_py.f90

MODULE INIT_PY

  USE data_py

CONTAINS

  SUBROUTINE burn_get_film_coeff_1d(bp,p_coor,Ts,Teuler,P_in,Qc,Qcprime)
  SUBROUTINE burn_init_1d(bp,bflag,Pin,To,rhoc,p_coor,rb,Toa,fr,Tn,Tflame)
```

`burn_get_film_coeff_1d(bp,p_coor,Ts,Teuler,P_in,Qc,Qcprime)`

- Variable `P = P_in`

- Variable `Te = min(Teuler,2*Tstar0)`, where `Tstar0` is read from the input file (Note: `Teuler` is T_g in Section 3)
  - Avoid Euler temperatures larger than twice the adiabatic flame temperature
  - Note: "This is a temporary hardwired fix. It should never affect the solution."
- Review `ixsymm` parameter read from the input file, and if it is nonzero, the cell is already burning at $t = 0$
  - If `ixsymm.ge.1`, then...
    - Variable `x_surf = p_coor(ixsymm)`, where `ixsymm` is 1, 2, or 3 for x, y, or z, respectively
    - Calculate the film coefficient h based on eqs (3.8) – (3.10)
    - Variable `front_dist = x_surf - x_surf_burn` where `x_surf_burn` is read from the input file
    - Variable `try_film` is determined from calculated h (and other constants)
  - Else...
    - Variable `try_film = film_cons`, which is read from the input file
- Calculate the heat flux `Qc` and the derivative of the heat flux `Qcprime` (Note: The heat flux is specified by $g(T_s, T_g)$ in Section 3.2)
  - Q_c (i.e., g) is given by eq (3.7) and cannot be negative, where the ΔT coefficient is the appropriate `try_film`
  - Q_c' (i.e., g') is given by eq (3.11), using the appropriate `try_film`

`burn_init_1d(bp,bflag,Pin,To,rhoc,p_coor,rb,Toa,fr,Tn,Tflame)`

- Review the `ixsymm` parameter read from the input file, and if it is nonzero, the cell is already burning at $t = 0$
  - If `ixsymm.ge.1`, then...
    - Variable `x_surf = p_coor(ixsymm)`, where `ixsymm` is 1, 2, or 3 for x, y, or z, respectively
    - If `x_surf.le.x_surf_burn`, then burn flag `bflag = 1`; else `bflag = 0`
  - Else...
    - Flag `bflag = 0`
- Convert `P = Pin*pa2atm`
- Calculate T_s and r_b given `To` and `P`
  - If `bflag.ne.0`, then...
    - If no table is read from input, then...
      - Calculate `rb` using eq (3.5)
      - Eq (3.6) is rearranged to solve for T_s
      - Using value for `rb`, calculate `Ts` from rearranged eq (3.6)
      - Variable `alp = alfac`, which is read from the input file

        * Else...
            · Call `polin2` (several times) and pass in information read from the table
        * Variable `xcond = rb / alp`
        * Variable `dtemp = Ts - To`
        * Loop over all grid points and calculate variable `Tn` (T^n) at every grid point for use later with eq (4.3)
        * Variable `Tflame = Tstar0`, which is read from the input file
    - Else...
        * Variable `dtemp = Tsurf - To` (Note: `Tsurf` **<u>is not</u>** the same as `Ts`!)
        * Loop over all grid points and calculate variable `Tn` at every grid point using linear scaling
        * Variable `Tflame = Tsurf`, which is read from the input file
        * Variable `rb = 0`

- Variable `Toa = Tsurf`, which is read from the input file
- Variable `fr = 0`
- Convert `rb = rb / m2cm`

5.3 Parameter Update at Time *t*

```
                              update_py.f90

MODULE UPDATE_PY

  USE data_py

CONTAINS

  SUBROUTINE BURN_GET_BURNING_RATE1D(bp,delt,P,To,Tn,Qc,Qc_old,Qr,Qr_old,
                                     rhoc,Toa,rb,fr,bflag,Tnp1,Tflame,coor)
  SUBROUTINE GFUN(bp,bw)
  SUBROUTINE MFUN(bp,bw)
  SUBROUTINE TFUN(bp,bw)
```

`BURN_GET_BURNING_RATE1D(bp,delt,P,To,Tn,Qc,Qc_old,Qr,Qr_old,rhoc,Toa,rb, fr,bflag,Tnp1,Tflame,coor)`

- If `press_min < P < press_max`, then proceed; else, print an error message and abort; where `press_min` and `press_max` are read from the input file
- Convert `P = P*pa2atm`
- Convert `rhoc = rhoc*kgmc2gcc`

- Convert `Qc = Qc*j_msq2cal_cmsq`
- Convert `Qcprime = Qr*j_msq2cal_cmsq`
- Variable `Ts = Tn(1)`
- Variable `To = To`, which is read from the input file
- Variable `lamc = rhoc*alfac*C`, where `alfac` and `C` are read from the input file
- Variable `nx = numx`, which read from the input file
- Call `MFUN(bp,bw)`
- Merge the previous r_b value with the new r_b value
- Loop over all grid points and calculate variable `Tnp1` (T^{n+1}) at every grid point using eq (4.3)
- Variable `Tnp1(nx) = To`, which is read from the input file
- Variable `ignited = Ts > Tignition`, where `Tignition` is read from the input file
- Call `GFUN(bp,bw)`
- Update the surface boundary conditions for the iteration
  - If the cell has ignited, then call `TFUN(bp,bw)`
  - Else, variable `Tstar = Tstar0`, which is read from the input file
- Variable `Tnp1(1) = Ts`
- Call `MFUN(bp,bw)`
- Merge the previous r_b value with the new r_b value
- Convert r_b back to MKS units
- Variable `Tflame = Tstar`
- Check output for divergence
  - If `rb_min < rb < rb_max`, then write it to the screen; else, print an error message and abort; where `rb_min` and `rb_max` are read from the input file
  - If `Tf_min < Tflame < Tf_max`, then write it to the screen; else, print an error message and abort; where `Tf_min` and `Tf_max` are read from the input file

`GFUN(bp,bw)`

- Solve for $(T_s^0)^{-1}$ by combining eqs (3.5) and (3.6)
- If parameter `ignited.eq.true`, then ...
  - If no information is read from the table, then...
    - Solve for variable `Tstar` ($T_\star$) using eq (3.12)
    - If `Ts.gt.Ts0.and.Ts.gt.Tslimit`, then `Tstar = Tstar0`, where `Tstar0` is read from the input file

    * Solve for variable `fs`, i.e., g, using eq (3.13)
    * Solve for variable `fsprime`, i.e., $\partial g/\partial T$, using eq (3.14)
  - Else...
    * Call `polin2` (several times) and pass in information read from the table
- Else...
  - Variable `fs = Qc / lamc`
  - Variable `fsprime = Qcprime / lamc`

`MFUN(bp,bw)`

- Calculate mass flux over density
- If no table is read from input, then...
  - Calculate `rb` from eq (3.6)
  - Variable `alfa_eff = alfac`, which is read from the input file
- Else...
  - Call `polin2` (several times) and pass in information read from the table

`TFUN(bp,bw)`

- Calculate the temperature of the injected gas, i.e, flame temperature, `Tstar`
- If no table is read from input, then...
  - Rearrange eq (3.12) and solve for `Tstar`
  - If `Ts.gt.Ts0.and.Ts.lt.Tslimit`, then `Tstar = Tstar0`, which is read from the input file
- Else...
  - Call `polin2` and pass in information read from the table

6 Input Control File and Using *Rocburn_PY*

Two example versions of the *Rocburn_PY* input file (RocburnPYControl.txt) are shown below. The first is a table that groups the input parameters by similarity as well as provides other information, such as the variable mapping between the theory (Section 3) and the program (Section 5). The second version shows a real, working *Rocburn_PY* problem for the NAWC #13 motor. In either version, note that some parameters refer to the solid (condensed) phase and some to the gas phase.

6.1 Summary of Input Parameters

Annotated *Rocburn_PY* Input (Control) Parameters

Note: The parameters given below have been rearranged by group according to similarity. This format **cannot** be used with as an actual *Rocburn_PY* input file because the code requires a specific ordering. A working *Rocburn_PY* problem is given in Section 6.2.

Input	Usage		Description	Units	Range (Typical)	
Empirical burning rate (power law) model parameters						
`a_p`	A_p	eq (3.5)	steady state regression speed at reference pressure	cm/s	$A_p > 0$	(0.4)
`n_p`	n_p	eq (3.5)	pressure exponent	none	$0 - 10$	(0.5)
`Pref`	P_{ref}	eq (3.5)	reference pressure	atm	$0 - 200$	(35)
Combustion (pyrolysis) model parameters						
`Ac`	A_c	eq (3.6)	pre-exponential factor	cm/s	$A_c > 100$	(2e5)
`eg_ru`	E_g/R_u	eq (3.12)	scaled activation energy for gas phase (i.e., gas temperature)	K	$E_g/R_u > 100$	(2.5e4)
`ec_ru`	E_c/R_u	eqs (3.6)	scaled activation energy for solid phase (i.e., solid temperature)	K	$E_c/R_u > 100$	(1.3e4)
`alfac`	α_c	eq (3.13)	condensed phase thermal diffusivity	cm^2/s	$0 - 1$	(2e-3)
`C`	c_p	eq (3.1)	specific heat of the solid	cal/(g K)	$0 - 1$	(0.4)
`lamg`	λ_g	eq (3.10)	gas phase thermal conductivity	cal/(cm s K)	$0 - 1$	(2e-4)
`delt`	Δt	eq (4.3)	simulation timestep	s	none	(3e-6)
`Tstar0`	$T^0_\star$	eq (3.12)	steady state (adiabatic) flame temperature	K	$T_{\text{ignition}} - 10,000$	(2500)
`To`	T_∞	eq (3.13)	supply (ambient) temperature	K	$100 - 1000$	(298)
Ignition model parameters						
`Tignition`	none	none	ignition temperature	K	$T_{\text{ignition}} > T_\infty$	(850)
`Tsurf`	none	none	surface temperature	K	$100 - T^0_\star$	(300)
`film_cons`	h/λ_c	eq (3.7)	film constant to determine heat flux in non-burning cells	$\text{W}/(\text{m}^2\,\text{K})$	none	(550)
`ixsymm`	none	none	enable/disable (axisymmetric) burning in cells at $t = 0$	none	$0 - 3$	(0)
`x_surf_burn`	none	none	width of propellant burning at $t = 0$	m	none	(0.2)
Ranges for pressure, temperature, and burn speed (divergence boundaries)						
`press_min`	none	none	minimum pressure allowed	Pa	none	(1e3)
`press_max`	none	none	maximum pressure allowed	Pa	none	(1e8)
`Tf_min`	none	none	minimum `Tflame` temperature allowed, where T_{flame} is the temperature of injected gas ($T_\star$)	K	none	(290)
`Tf_max`	none	none	maximum `Tflame` temperature allowed, where T_{flame} is the temperature of injected gas ($T_\star$)	K	none	(1e4)

continued on next page...

...*continued from last page*

Input	Usage		Description	Units	Range (Typical)	
`rb_min`	none	none	minimum burn rate allowed	m/s	none	(-1e-9)
`rb_max`	none	none	maximum burn rate allowed	m/s	none	(1e2)
			Grid parameters			
`igrid`	none	none	flag to specify computational grid type	none	none	(2)
`xmax`	none	none	width into propellant depth of interface region	m	$x_{\max} < 0$	(-0.2)
`numx`	none	none	maximum number of $x_{\max}$ grid points	none	none	(100)
`beta`	none	none	grid stretch constant used to resolve heat flux at surface	none	none	(1.01)
			***Enable lookup table to compute the flux** g*			
`TABUSE`	none	none	enable/disable lookup table	none	none	(0)
`TABNAM`	none	none	name of the file with table	none	none	(none)

As shown above, the list of required input parameters is extensive, 29 in total. Some of them are straightforward, especially the burn parameters that can be taken from literature values. There are others, however, that require more in depth discussion.

6.1.1 Combustion Model Parameters

While the combustion model is well-documented, selecting an appropriate timestep, `delt`, is less so. Since a first-order integrator is used (Section 4), the timestep must be small, so the integrator can be fast. As a result it is important to watch the numerical stability associated the selected timestep.

6.1.2 Ignition Model Parameters

The ignition model in *Rocburn_PY* can be confusing, since a number of the parameters have similar names and are not necessarily specifically addressed by the theory presented in Section 3. For this reason, a simple schematic of a "standard rocket" is given in Figure 6.1 to aid in the presentation.

Figure 6.1 shows a simple rocket with the nozzle at one end (left), the tail at the other (right), and the propellant in the middle. In the center of the chamber is a bore through which the gas escapes as the rocket (propellant) burns away. Depending on the situation, the propellant may or may not be burning at $t = 0$ (orange vs gray propellant, respectively), and `x_surf_burn` (i.e., the value of x that represents the burning surface) distinguishes between burning and non-burning propellant.

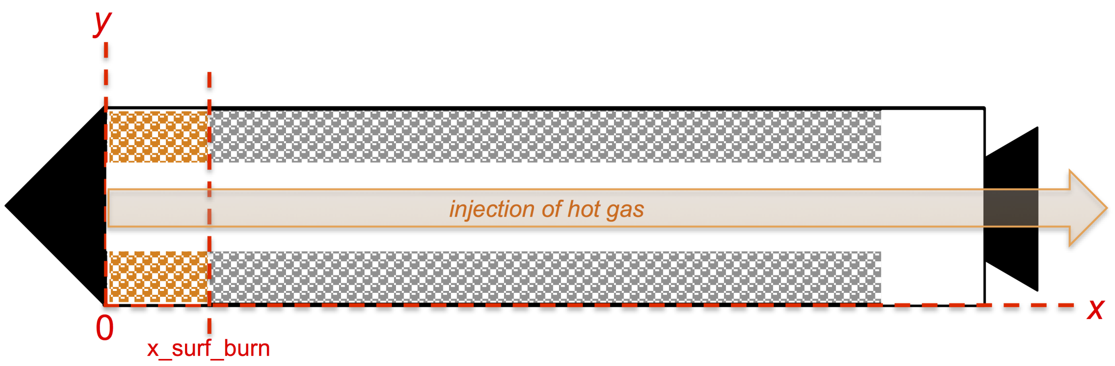


Figure 6.1: Simple rocket. This schematic of a "rocket" illustrates the different regions, from the nozzle (far left), to the propellant (center), to the tail (far right). The variable `x_surf_burn` distinguishes between burning and non-burning propellant.

film_cons The input parameter `film_cons` is somewhat misleading as it sounds like the film *coefficient*, h, such as in Section 3.2.1. In actuality it is a combination of two constants.

$$\mathcal{C}_{\text{film}} = \frac{h}{\lambda_c} = \frac{0.0287\rho_g U_\infty c_p}{\lambda_c}\left(\frac{x\rho_g U_\infty \mu_g c_p^2}{\lambda_g^2}\right)^{-1/5} \tag{6.1}$$

where λ_c is the thermal conductivity of the solid. (See eq (3.10).) As illustrated in Section 5.1, `film_cons` is used to calculate $g(T_s, T_g)$ in eq (3.7) and $\partial g/\partial T_s$ in eq (3.11).

ixsymm This parameter is a flag to enable or disable (axisymmetric) burning in cells at $t = 0$. This parameter can take on one of four values:

0 No burning at $t = 0$

1 Burning along x-direction at $t = 0$

2 Burning along y-direction at $t = 0$

3 Burning along z-direction at $t = 0$

x_surf_burn This parameter is intimately tied to `ixsymm` as it specifies the width (or distance) of the propellant that is burning at $t = 0$, such as that shown in Figure 6.1. Note that if `ixsymm = 0`, then there is no propellant burning when the simulation begins (i.e., at $t = 0$), and `x_surf_burn` is unused.

6.1.3 Divergence Boundaries

Rocburn_PY will stop if the values for P, T_{flame} (i.e., $T_\star$), and r_b fall outside of their user-specified ranges.

- $P_{\text{min}} < P < P_{\text{max}}$
- $T_{\text{min}} < T_\star < T_{\text{max}}$
- $r_{b,\text{min}} < r_b < r_{b,\text{max}}$

There are old versions of the *Rocburn_PY* input file that did not require these ranges to be specified in the input file. If these old versions are used, *Rocburn_PY* will use internally set limits.

- $1\,\text{kPa} < P < 100\,\text{MPa}$
- $290\,\text{K} < T_\star < 10\,000\,\text{K}$
- $-1\,\text{nm/s} < r_b < 100\,\text{m/s}$

6.1.4 Grid Parameters

As discussed in the introduction to Section 5, *Rocburn_PY* specifies a very thin temperature boundary layer associated with the heat wave traveling into the solid. More importantly this thin region requires a stretched grid to properly resolve the heat flux at the surface.

igrid This parameter is a flag to select the type of computational grid used to represent this thin region. This parameter can take on one of two values:

1 Exponential grid

2 Boundary layer grid

xmax|numx This parameter specifies the width of this thin region, i.e., the width into propellant depth of interface. Note that `xmax` must be negative! In addition the density of points in this region is specified by `numx`, i.e., the maximum number of `xmax` grid points. These two parameters are intimately tied to `igrid`. If an exponential grid is selected, i.e., `igrid = 1`, there two parameters are unused.

beta As suggested above, the interface region requires a slightly different grid to properly resolve the heat flux at the surface. The grid parameters are scaled by the stretch constant `beta`.

6.1.5 Lookup Table Parameters

As briefly touched upon in Section 1 and again at the end of Section 3.2.2, *Rocburn_PY* has a unique feature that allows for use of an externally calculated parameter table. This table is created by averaging the solid phase heat conduction over a plane that is parallel to the mean surface, accounting for the rotational terms that arise due to the uneven surface.

TABUSE|TABNAM The `TABUSE` parameter is a flag to enable or disable the use of an external table. This parameter can take on one of two values:

- **0** Use analytical results to calculate properties and parameters (i.e., no table)
- **1** Exercise the table algorithm portion of *Rocburn_PY*

If `TABUSE = 1` *Rocburn_PY* will read in the information from a text file, the name of which is specified by `TABNAM`, e.g., RBRNtable.dat. Note that if `TABUSE = 0`, `TABNAM` is unused.

6.2 Working *Rocburn_PY* Input File

A working example of the *Rocburn_PY* input file (RocburnPYControl.txt) is shown below, i.e, the input parameters are in the correct order as required by *Rocburn_PY*. Note that the line numbers at the left of this sample file (e.g., 1, 2, 3, etc.) are **not** part of the input file; they are provided for ease of reference. (See Section 5.)

```
                              RocburnPYControl.txt

 1   0.3912       a_p          in rb = a_p*(P/Pref)^n, rb in cm/sec & P in atm
 2   0.461        n_p          in rb = a_p*(P/Pref)^n, rb in cm/sec & P in atm
 3   34.0         Pref         in rb = a_p*(P/Pref)^n, atm
 4   180000.0     Ac           Condensed_phase_prefactor,(cm/s)
 5   24000.0      eg_ru        Gas phase activation temp, (K)
 6   12500.0      ec_ru        Solid phase activation temp, (K)
 7   1.00e-3      alfac        Solid phase thermal diffusivity, (cm^2/s)
 8   0.350        C            Specific heat (ga & condensed phases), (cal/g-K)
 9   2.00e-4      lamg         Gas phase thermal conductivity, (cal/cm-s-K)
10   1.0e-6       delt         Timestep, (s);_WATCH STABILITY
11   2            igrid        Grid control distribution; 1 = exp, 2 = bl
12   100          numx         number of points in propellant depth
13   -0.200       xmax         Maximum x location, (cm); MUST BE NEGATIVE!
14   1.010        beta         Grid stretching parameter,
15   2850.0       Tstar0       adiabatic flame temperature, Tstar0 (K)
16   300.0        To           cold temperature, To (K)
17   850.0        Tignition    ignition temperature, Tignition (K)
18   300.0        Tsurf        surface temperature
19   560.08d0     film_cons    constant in film coefficient [ W/ (m^2 K) ]
20   1            ixsymm       axisymmetric initial burning, use x_surf_burn
21   0.1815       x_surf_burn  last surface x location burning from the onset
22   1.d8         press_max    maximum pressure allowed to be passed in [Pa]
23   1.d2         press_min    minimum pressure allowed to be passed in [Pa]
24   1.d0         rb_max       maximum burn rate allowed [m/sec]
25   -1.d-6       rb_min       minimum burn rate allowed [m/sec]
26   1.d5         Tf_max       maximum gas temperature allowed [K]
27   100.00       Tf_min       minimum gas temperature allowed [K]
```

```
28   0             TABUSE        1 USE table algorithm, 0 USE analytical results
29   RBRNtab.dat   TABNAM        name of the file w/ table
```

References

Blomshield, F. S., Mathes, H. B., Crump, J. E., Beiter, C. A., and Beckstead, M. W. (1997). Nonlinear stability testing of full-scale tactical motors. *J. Propul. Power*, 13:350–366.

Brewster, M. W., Ward, M. J., and Son, S. F. (2000). Simplified combustion modeling of double base propellant: Gas phase chain reaction vs thermal decomposition. *Combust. Sci. and Tech.*, 154:1–30.

Chen, M., Buckmaster, J., Jackson, T. L., and Massa, L. (2002). Homogenization issues and the combustion of heterogeneous solid propellants. *Proc. Combust. Inst.*, 29:2923–2929.

Huggett, C., Bartley, C. E., and Mills, M. M. (1960). *Solid Propellant Rockets*. Princeton University Press, Princeton.

Ibiricu, M. M. and Williams, F. A. (1975). Influence of externally applied thermal radiation on the burning rates of homogeneous solid propellants. *Combust. Flame*, 24:185–198.

Jackson, T. L. and Buckmaster, J. (2002). Heterogeneous propellant combustion. *AIAAJ*, 40:1122–1130.

Kovalskii, A. A., Konev, E. V., and V.Krasilnikov, B. (1967). Combustion of nitroglycerine powder. *Fiz. Goreniya Vzryva*, 3:547–554.

Massa, L., Jackson, T. L., and Buckmaster, J. (2004). Using heterogeneous propellant burning simulations as subgrid components of rocket simulations. *AIAAJ*, 42:1889–1900.

Massa, L., Jackson, T. L., and Buckmaster, J. (2005). New kinetics for a model of heterogeneous propellant combustion. *J. Propul. Power*, 21:914–924.

Massa, L., Jackson, T. L., Buckmaster, J., and Campbell, M. (2002). Three-dimensional heterogeneous propellant combustion. *Proc. Combust. Inst.*, 29:2975–2983.

Ward, M. J., Son, S. F., and Brewster, M. Q. (1998). Steady deflagration of hmx with simple kinetics: A gas phase chain reaction model. *Combust. Flame*, 114:556–568.

Williams, F. A. (1973). Quasi-steady gas-phase flame theory in unsteady burning of a homogeneous solid propellant. *AIAAJ*, 11:1328–1330.