Rocburn2D User's and Developer's Guide

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign
2260 Digital Computer Laboratory
Urbana, IL

Title:	Rocburn2D User's and Developer's Guide
Author:	X. Jiao, L. Massa, K. Tang
Subject:	
Revision:	1
Revision History	Revision 1: Initial Release
Effective Date:	3/17/2005

Table of Content

1 Introduction

1.1 Purpose

Rocburn2D is part of the combustion module *Rocburn*. The main purpose of *Rocbun2D* is to provide the operations (data transfer, memory allocation, communication with manager code, calling one-dimensional combustion routines, sub-cycling, and so on) at the two-dimensional level on the burning surface while the burning rate of a single point on the burning surface is calculated by one of the one-dimensional combustion routines provided. This implementation supports the plug-and-play (modular) capability in the combustion module (*Rocburn*) implemented in the integrated system code (currently *GEN3*). The system code user can chose the combustion model used in the simulation by specifying the module_name when calling the subroutine ROCBURN_LOAD_MODULE (see *GEN3* User's Guide [1] for more details about how to select 1-D combustion model in the numerical simulation). The added benefit is that the integration of additional one-dimensional combustion models into the system code will not require any changes in the previously implemented one-dimensional combustion models (at least in principle). The implementation of new combustion model does not need to change the operations in two-dimensional level (e.g. on the burning surface) as long as the new combustion model provides the mandatory subroutines complied with the interface common to all one-dimensional combustion models as described in Section 3.2. Hopefully only minor changes in Rocburn_load_module.f90 and Makefile are needed when adding new combustion model (see Section 7).

1.2 Overview

Rocburn is a combustion module to provide the regression rate of solid propellant ($\dot{r}_b$) at the burning propellant surface required in the mass conservation and energy conservation at the solid-fluid interface as (adapted from *GNE3* Developers' Guide [2]):

Mass conservation:

$$\vec{v}_f \cdot \vec{n} = -\frac{\dot{m}}{\rho_f} + (\vec{v}_s + \dot{r}_b \vec{n}) \cdot \vec{n}$$

Momentum conservation:

$$\vec{t}_s = \vec{t}_f + \dot{m}(\vec{v}_s - \vec{v}_f)$$

where

$\dot{m}$ = mass flux
$\dot{r}_b$ = burning rate
ρ = density

$\vec{n}$ = normal direction vector on the deformed propellant surface measured positive into solid
$\vec{t}$ = traction vector
$\vec{v}$ = velocity vector

The subscripts *s* and *f* denote solid and fluid respectively.

Based on above mentioned mass and momentum conservation at the fluid-solid interface, the incoming and outgoing data for combustion module (*Rocburn*) handled by *Rocburn2D* during time marching are:

Incoming data:

$(p_f)_{FF}^{n+\alpha}, (q_c)_{FF}^{n+\alpha}, (q_r)_{FF}^{n+\alpha}, (\rho_s)_{FF}^{n+\alpha}, (T_f)_{FF}^{n+\alpha}$, and $(\vec{y})_{FN}^{n+\alpha}$ (for visualization and/or convective heat flux calculation)

Outgoing data:

$(\dot{r}_b)_{FF}^{n+1}, (T_{flame})_{FF}^{n+1}$, and $(\text{bflag})_{FF}^{n+1}$

[Note: Before ignition, T_{flame} is propellant surface temperature. During initialization, $(\text{bflag})_{FF}^{n+1}$ is an incoming buffer from fluids.]

where

p_f = pressure
q_c = convective heat flux
q_r = radiative heat flux
ρ_s = propellant density
T_{flame} = flame temperature
$\vec{y}$ = deformed location vector (for visualization and/or empirical convective heat flux calculation)
bflag = burn flag
$(\cdot)_{FF}$ = value computed at faces of the fluid interface
$(\cdot)_{FN}$ = value computed at nodes of the fluid interface

The superscript *n* denotes current time step, *n+1* denotes next time step, and $n+\alpha$ denotes the local time step during sub-cycling when interpolation of incoming data become necessary. The local time index α is defined as

$$\alpha = (t - t^n) / \Delta t^n$$

Since the data are transferred between component codes only at every "system time step" Δt, sub-cycling may become necessary in *Rocburn* when $\Delta t_B < \Delta t$.

The initial value of `bflag` is setup in fluid code with `bflag = 1` indicating the propellant is ignited and `bflag = 0` indicating the propellant is not ignited for the ignitable panes initially. *Rocburn* will initialize the combustion model and turn on/off ignition simulation according to the value of `bflag`. If the initial value of `bflag = 0`, ignition simulation will be turned on in

Rocburn. After the propellant is ignited, *Rocburn* will update the value of `bflag` from `bflag = 0` to `blfag =1`.

Current implementation of *Rcoburn* supports the coupling algorithm and the transfer of interface data between different components (namely *Rocflo*, *Rocflu*, *Rocfrac*, *Rocsolid*, *Rcoburn*, and many others) implemented in the *GEN3* integrated code. As described in *GEN3* User's Guide, *GEN3* supports the modular approach by moving all the interaction and interface specific operations from the physical components (combustion codes, fluid codes, and solid codes) to the interface and manager codes (*Rocman* and *Roccom*).

Similarly, the implementation of combustion module *Rocburn* also allows the modulation of different combustion models. To achieve this objective, implementation of *Rocburn* is consisted of a main driver (*Rocburn2D*) and a set of plug-in 1-D combustion model subroutines (*Rocburn_APN*, *Rcoburn_PY*, and *Rocburn_ZN* are currently available). The main functions of the driver routine, *Rocburn2D*, are (1) initialization and memory allocation at the two-dimensional level on the burning surface, (2) communicating with other components in the system code through *Roccom* and *Rocman*, and (3) calculating the burning rate for the entire burning surface by calling the selected 1-D combustion model for which the burning characteristics for a single point on the propellant surface is needed. Currently three combustion models are available: *Rcoburn_APN*, *Rocburn_PY*, and *Rocburn_ZN*. A schematic diagram for *Rcoburn* implementation is shown in Figure 1. See Section 4 for a brief discussion of these models. In summary, *Rocburn_APN* is a quasi-steady combustion model using the Vieille's or Saint Robert's law ($\dot{r}_b = aP^n$) while *Rocburn_PY* and *Rocburn_ZN* are unsteady combustion models. For detailed information about the unsteady models, reader should refer to the user's and developer's guide for *Rcoburn_PY* [3, 4] and *Rocburn_ZN* [5, 6]. The rest of this document will be focused on *Rocburn2D* and the requirements for implementing 1-D combustion model.

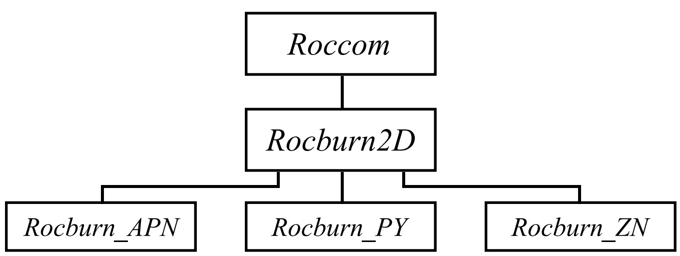

Figure 1. *Rocburn* code structure.

2 *Rocburn2D* Variables

Let β be a surface grid index and γ be an index to a fixed set of points inside the propellant. The number of points on the surface $n(\beta)$ is determined by grid requirements set elsewhere[1], while the number of points inside the propellant, i.e. surface $n(\gamma)$, is specified by each individual 1-D model and has, in principle, no relation with the geometrical specification of the problem. A list of the 2D variables used by *Rocburn* is reported below.

Variable	Intention	Description	Type	Unit
$T_c(\beta, \gamma)^n$	IN	Condensed Phase Temperature	DBL, DIM(:,:)	K
$T_c(\beta, \gamma)^{n+1}$	OUT	Condensed Phase Temperature	DBL, DIM(:,:)	K
$r_b(\beta)^{n+\alpha}$	INOUT	Burning rate	DBL, DIM(:)	m/s
$T_{oa}(\beta)^{n+\alpha}$	INOUT	Apparent initial temperature	DBL, DIM(:)	K
$T_\infty(\beta)^{n+\alpha}$	IN	Initial temperature	DBL, DIM(:)	K
$q_c(\beta)^{n+\alpha}$	IN	Convective heat flux	DBL, DIM(:)	W/cm^2
$q_r(\beta)^{n+\alpha}$	IN	Radiant heat flux	DBL, DIM(:)	W/cm^2
$q_c(\beta)^{n-1+\alpha}$	IN	Convective heat flux	DBL, DIM(:)	W/cm^2
$q_r(\beta)^{n-1+\alpha}$	IN	Radiant heat flux	DBL, DIM(:)	W/cm^2
$P(\beta)^{n+\alpha}$	IN	Pressure	DBL, DIM(:)	Pa
$\rho_c(\beta)^{n+\alpha}$	IN	Propellant density	DBL, DIM(:)	Kg/m^3
$T_{flam}(\beta)^{n+1}$	OUT	Flame temperature	DBL, DIM(:)	K
$f_r(\beta)^{n+\alpha}r$	INOUT	Fraction of radiation absorbed below surface reaction zone	DBL, DIM(:)	-
$T_{gas}(\beta)^{n+\alpha}$	IN	Gas temperature (when fluid codes can not supplied convective flux during ignition simulation)	DBL, DIM(:)	K
$xyz(\beta)^{n+\alpha}$	IN	Deformed location vector $\bar{y}$	DBL, DIM(:,3)	M
blag$(\beta)^{n+\alpha}$	INOUT	Burn flag	INT, DIM(:)	-

These variables are specified in MODULE M_ROCBURN_INTERFACE_DATA and the whole dataset is defined as derived type `block` as (see file: 01.burn_interface_data.f90 in directory `rocstar/Rocburn_2D/Codes`):

```
TYPE, PUBLIC :: block
```

[1] Under current implementation in *GEN3*, this is restricted to be inheriting only from the ignitable panes of the fluid interface. Plan has been made to allow *Rocburn* to inherit mesh from the ignitable panes of either fluid or solid interface. Most of the revisions related to this planned changes will be made in *Rocman*.

```
!
!          data for initialization
!

     INTEGER                    :: iblock           ! block id
     INTEGER                    :: nfaces           ! number of faces




!
!  ---------------------------------------------
!       incoming data from ROCMAN
!
     REAL(DBL), POINTER         :: coor(:,:)
     REAL(DBL), POINTER         :: pres(:)
     REAL(DBL), POINTER         :: qr(:), qc(:)     ! not used for APN
     REAL(DBL), POINTER         :: rhoc(:)          ! not used for APN
!    REAL(DBL), POINTER         :: To(:)            ! from solid if available

!       incoming data for specific burning rate model

     REAL(DBL), POINTER         :: Tg(:)            ! Rocburn_PY
     INTEGER, POINTER           :: burn_flag(:)     ! Rocburn_PY
!
!
!  ---------------------------------------------
!       outgoing data to ROCMAN
!
     REAL(DBL), POINTER         :: rb(:)
     REAL(DBL), POINTER         :: Tf(:)

!  ---------------------------------------------
!
!       internal data storage for burning rate models
!
!
!          independent variables; old state variables
!
     REAL(DBL), POINTER         :: qc_old(:), qr_old(:)
     REAL(DBL), POINTER         :: pres_old(:), Tg_old(:)
     REAL(DBL), POINTER         :: rhoc_old(:)
!    REAL(DBL), POINTER         :: To_old(:)

!
!          dependent variables
!

     REAL(DBL), POINTER         :: temp(:,:)
!
!          data storage for specific burning rate model --- dependent varialbes
!


     REAL(DBL), POINTER         :: Toa(:)           ! Rocburn_ZN
     REAL(DBL), POINTER         :: fr(:)            ! Rocburn_ZN
     REAL(DBL), POINTER         :: dist(:)          ! Rocburn_PY

  END TYPE block
```

The data transfer between *Rocburn2D* and *Rocman* should be passing though the dummy arguments during subroutine calls only. A pointer G_b is used in order to achieve this goal. The pointer G_b is the linked list that is built based on the above mentioned derived type: `block` for the cases of multiple bocks in one processor. Allocation of G_b is executed in subroutine ROCBURN_LOAD_MODULE. **One-dimensional combustion model should not use this module.** Communications between *Rocburn2D* and 1-D combustion models are conducted using another pointer G_1d (the dataset defined by individual combustion model that is specific to the individual model) passing as argument (see section 3.3 for detail).

2.1 Notes

- Local time index α in the variable list should be equal to 1 if the heat flux and mass flux balance is imposed at the end of the system time step. Since *Rocman* uses linear interpolation to calculate the value at time level $n+\alpha$, *Rocburn2D* does linear interpolation internally when the sub-cycling is needed.

- In case of a Navier-Stokes simulation in which the thermal boundary layer (TBL) is resolved, the convective heat flux to the surface is computed in fluid codes and passed to *Rocburn* as an incoming data. When, conversely, an Euler simulation is performed, or the grid is too coarse to resolve the TBL, the convective heat flux can be evaluated by *Rocburn (*available in *Rocburn_PY)* using empirical formulation. The subroutine filmcoeff provides this capability. Therefore, for the case of Euler simulations in fluids or the case of no convective heat flux (q_c) available from fluids, two extra variables, the gas temperature, T_{gas}, and the mesh physical coordinates, xyz, need to be provided as incoming buffer. The flag `G_b%TBL_flag` is used to indicate whether the thermal boundary layer is resolved in fluids or not. The value of `G_b%TBL_flag` is designed to be determined by fluids and sent to *Rocburn*. However, in current implementation, the value of `G_b%TBL_flag` is not transmitted from fluids but is set to zero (e.g. thermal boundary layer is not resolved in fluids) in subroutine ROCBURN_LOAD_MODULE (see file Rocburn_load_module.f90). In another word, *Rocburn* always uses subroutine filmcoeff to estimate connective heat flux from chamber to solid propellant surface in current implementation. This implementation should be revised if the connective heat flux is available from fluids in the future.

- Variables used in *Rocburn2D* can be divided into two groups: variable that need to be kept from iteration to iteration, (state variables), and variables that might be overridden (buffer variables). These represent incoming and outgoing buffers between *Rocburn2D* and other modules. In principle an incoming buffer can be re-used as outgoing buffer; this would reduce memory allocation. For example, the flame temperature, T_{flame}, is always an outgoing buffer, while the Euler temperature at the wall, T_{gas}, is always an incoming buffer; the same physical memory registers could be used for both variables.

- Values of the convective and radiant heat fluxes at consecutive times, q_c and q_r, need to be provided. This necessity arises from the use of a Newton method to handle the nonlinear combustion equations. Whether values at two time levels are required for both the heat fluxes or for the radiative heat flux only, is to be reconsidered.

- Global parameters of the combustion model are included in separate modules for different combustion models (*Rocburn_APN*, *Rocburn_PY*, and *Rocburn_ZN*). The module (i.e. M_Rocburn_1D_APN, M_Rocburn_1D_PY, and M_Rocburn_1D_ZN) is specific to each specific burn rate model and should not share information with *Rocburn2D*. Moreover all the variables included in such module need to be constant at all times. This to comply with ``no global variable can change requirement'' imposed by CHARM++. All the variables that can change in time should be passed through subroutine arguments.

3 Implementation

To comply with *GEN3* requirement, all the interface specific operations have been moved to *Roccom* and *Rocman* including restart, interface data input/output, handling of predictor-corrector iterations, interpolation of data required during system level sub-cycling, and so on. *Rocburn* now provides the following functions:

- Initialize *Rocburn* including inheriting the ignitable panes from fluids, opening windows, initializing data structure, allocating memory, initializing the 1-D combustion model selected, and registration with *Roccom*.

- Update *Rocburn* solution during time marching; support the control flow required by system code including:

  (a) Obtains $(p_f)_{FF}^{n+\alpha}, (q_c)_{FF}^{n+\alpha}, (q_r)_{FF}^{n+\alpha}, (\rho_s)_{FF}^{n+\alpha}, (T_f)_{FF}^{n+\alpha}$, and $(y)_{FN}^{n}$ from *Rocman* by extrapolation

  (b) Update solution of get $(\acute{Y}_b)_{FF}^{n+\alpha}$ *and* $(T_{flame})_{FF}^{n+\alpha}$

  (c) At the end of sub-cycling, sends $(\acute{Y}_b)_{FF}^{n+1}$ *and* $(T_{flame})_{FF}^{n+1}$ to *Rcoman.*

- Finalize *Rocburn* including deleting windows and de-allocating memory.

3.1 *Rocburn2D* subroutines

The requirements for physical components were detailed in the *GEN3* Developer's Guide [2]. In this section, specific requirements for *Rocburn2D* are described. *Rocburn2D* needs to provide and register with *Roccom* the following subroutines (in file rocburn_2D.f90):

```
INIT_WRAPPER( G_b, initial_time, comm, MAN_INIT, inSurf, inINT, IN_obt_attr)
```

Variable	Intention	Description	Type	Unit
G_b	-	Pointer to *Rocburn2D* variables	TYPE list_block	-
initial_time	IN	Initial time	DBL	s
comm	IN	Communicator	INT	-
MAN_INIT	IN	*Rocman* Handles (see *Rocman* user guide)	INT	-
inSurf	IN		CHAR	-
inInt	IN		CHAR	-
IN_obt_attr	IN	*Rocman* Handles (see *Rocman* user guide)	INTEGER	-

In `INITIALIZE`, the following tasks are included:

- Call INIT_0D to initialize 1-D combustion model at 0-D level
- Create interface data in *Roccom* and allocate memory for them (in G_b)
- Create internal data that need to be saved for predictor-corrector in *Roccom* and allocate memory for them
- Get size information from Roccom (number of blocks, block identifications, and size of block)
- Obtain memory address from *Roccom* and build up the blocks.
- Handle difference in *Rocburn2D* initialization scheme with and without restart depending on the value of initial_time.
- Call INIT_1D to initialize 1-D combustion model at 1-D level for all blocks and all cells

```
UPDATE_WRAPPER( G_b, timestamp, dt, MAN_UPDATE)
```

Variable	Intention	Description	Type	Unit
G_b	IN	Pointer to *Rocburn2D* variables	TYPE list_block	-
timestamp	IN	Time stamp	DBL	s
dt	IN	Size of time step	DBL	s
MAN_UPDATE	IN	*Rocman* Handles (see *Rocman* user guide)	INTEGER	-

In UPDATE, the following tasks are included:

- Provide special operations for the *Rocburn_APN* model
- Call calcdist_2D to calculate the distance from flame front and call GET_FILM_COEFF_1D to calculate convective heat flux if fluids could not provide convective heat flux (i.e. NO_TBL = 0)
- Call GET_TIME_STEP_1D to calculate time step for combustion module
- Perform sub-cycling if necessary
- Call GET_BURNING_RATE_1D to simulate ignition before propellant ignited (blfag = 0)
- Call GET_BURNING_RATE_1D to calculate solid propellant burning rate and flame temperature for a given set of pressure, radiant flux, initial cold temperature, and propellant density after propellant ignited (blfag = 1)

```
FINALIZE( G_b)
```

Variable	Intention	Description	Type	Unit
G_b	IN	Pointer to *Rocburn2D* variables	TYPE list_block	-

In case of convective heat flux not available from fluids, subroutine caldist_2D is used to calculate the distance from the flame front so that the heat transfer coefficient can be determined by GET_FILM_COEFF_1D.

```
CALCDIST_2D( G_b, xyz_2d, dist_2d)
```

Variable	Intention	Description	Type	Unit
G_b	IN	Pointer to *Rocburn2D* variables	TYPE list_block	-
xyz_2d	IN	Deformed location	DBL, DIM(:,3)	m
dist_2d	OUT	Distance from flame front	DBL, DIM(:)	m

3.2 Mandatory subroutines provided by one-dimensional combustion model

In order to support the modularity, a common interface for 1-D combustion model is designed. The following subroutines are required for 1-D combustion model so that the implementation of new combustion model will not require changes in *Rocburn2D*. The arguments appeared in the following subroutines are mandatory for all 1-D combustion models.

Subroutines for use in `INITIALIZE`

`INIT_0D( g_1d, comm, Indir, nxmax, To_read)`

Variable	Intention	Description	Type	Unit
g_1d	IN	Pointer to global variables of 1-D combustion model	TYPE G_BURN_1D	-
comm	IN	Communicator	INT	-
Indir	IN	Input file directory	CHAR	-
nxmax	OUT	Maximum number of grid points in propellant	INT	-
To_read	OUT	Initial propellant temperature	DBL	K

In `INIT_0D`, the following tasks are suggested:

- Allocate the 1-D combustion model global variables `g_1d`
- Input the propellant property parameters (in `g_1d`) used by 1-D combustion model
- Grid generation

`INIT_1D( g_1d, bflag, P, To, rhoc, p_coor, rb, Toa, fr, Tn, Tflame)`

Variable	Intention	Description	Type	Unit
g_1d	IN	Pointer to global variables of 1-D combustion model	TYPE G_BURN_1D	-
bflag	INOUT	Burn flag	INT	-
P	IN	Pressure	DBL	Pa
To	IN	Propellant initial/cold temperature	DBL	K
rhoc	IN	Propellant density	DBL	Kg/m^3
p_coor	IN	Coordinates	DBL, DIM(3)	m
rb	OUT	Burning rate	DBL	m/s

Toa	OUT	Apparent temperature	DBL	K
fr	OUT	Fraction of radiation absorbed below surface reaction zone	DBL	-
Tn	OUT	Propellant temperature profile	DBL, DIM(:)	K
Tflame	OUT	Flame temperature	DBL	K

In `INIT_1D`, the following tasks are suggested:

- Determine the initial conditions needed for 1-D combustion model with or without ignition simulation

Subroutines for use in `UPDATE`

```
GET_FILM_COEFF_1D( g_1d, p_coor, Ts, T_euler, P, Qc, Qcprime)
```

Variable	Intention	Description	Type	Unit
g_1d	IN	Pointer to global variables of 1-D combustion model	TYPE `G_BURN_1D`	-
p_coor	IN	Coordinates	DBL, DIM(3)	m
Ts	IN	Propellant surface temperature	DBL	K
T_euler	IN	Gas phase surface temperature	DBL	K
P	IN	Pressure	DBL	Pa
Qc	OUT	Approximated convective heat flux	DBL	W/m^2
Qcprime	OUT	Approximated derivative of convective heat flux with respect to temperature	DBL	$W/m^2/K$

In `GET_FILM_COEFF_1D`, the following tasks are suggested:

- Estimate convective heat flux from chamber gas
- Estimate the derivative of convective heat flux with respect to surface temperature (for *Rocburn_PY*)

```
GET_TIME_STEP_1D( g_1d, rb, Toa, dt_max)
```

Variable	Intention	Description	Type	Unit
g_1d	IN	Pointer to global variables of 1-D combustion model	TYPE G_BURN_1D	-
rb	IN	Burning rate	DBL	m/s
Toa	IN	Apparent temperature	DBL	K
dt_max	OUT	Maximum time step	DBL	s

In GET_TIME_STEP_1D, the following tasks are suggested:

- Estimate maximum size of time-step in combustion model in order to determine whether sub-cycling in *Rocburn2D* is necessary

```
GET_BURNING_RATE_1D ( g_1d, delt, P, To, Tn, qc, qc_old, qr, qr_old, rhoc, &
                      Toa, rb, fr, bflag, Tnp1, Tflame)
```

Variable	Intention	Description	Type	Unit
g_1d	IN	Pointer to global variables of 1-D combustion model	TYPE G_BURN_1D	-
delt	IN	System time step	DBL	s
P	IN	Pressure	DBL	Pa
To	IN	Propellant initial/cold temperature	DBL	K
Tn	IN	Propellant temperature profile at current time step	DBL, DIM(:)	K
qc	IN	Convective heat flux at current time step	DBL	W/m^2
qc_old	IN	Convective heat flux at next time step	DBL	W/m^2
qr	IN	Radiative heat flux at current time step	DBL	W/m^2
qr_old	IN	Radiative heat flux at next time step	DBL	W/m^2
rhoc	IN	Propellant density	DBL	Kg/m^3
Toa	INOUT	Apparent temperature	DBL	K
rb	INOUT	Burning rate	DBL	m/s
fr	INOUT	Fraction of radiation absorbed below surface reaction zone	DBL	-
bflag	INOUT	Burn flag	INT	-

Tnp1	OUT	Propellant temperature profile at next time step	DBL, DIM(:)	K
Tflame	OUT	Flame temperature	DBL	K

In `GET_BURNING_RATE_1D`, the following tasks are suggested:

- Perform convective (subject to `qc`) and/or radiant (subject to `qr`) ignition simulation if propellant has not ignited
- Calculate solid propellant burning rate `rb` and flame temperature `Tflame` for a given set of pressure `P`, radiant flux `qr`, solid propellant density `rhoc`, and initial temperature `To` if the propellant has ignited

3.3 Issue of the global variables defined by 1-D combustion model used in *Rocburn2D*

Global variables specific to 1-D combustion model need to be transferred between different subroutines provided by the 1-D combustion model. Different 1-D combustion model has different requirement for its global variables. It is inevitable for *Rocburn2D* to get access to 1-D global variable. *Rocburn2D* needs to refer to the global variables for 1-D combustion model. Hence, it appears that *Rocburn2D* need to have the information about all the global variables for all the 1-D combustion models provided. However in order to fully support plug-and-play modularity, totally separation between (1) the global variables used by *Rocburn2D* and the selected 1-D combustion model, and (2) the global variables used by the 1-D combustion models provided, is necessary. The goal is for *Rocburn2D* to refer to the specific 1-D global variables for the selected 1-D combustion model without knowing the other 1-D global variables related to the other 1-D combustion models not selected. To achieve this goal, a "workaround" algorithm has been implemented to deceive the F90 compiler to point the pointer to the global variables specific to the 1-D combustion model selected without knowing/referring/including other 1-D global variables used by un-selected combustion models. A placeholder for the derived type defined by 1-D combustion model is defined and used as:

```
! This is a placeholder for the type defined in 1D modules
  TYPE, PUBLIC :: G_BURN_1D
      INTEGER :: buf(4096)
  END TYPE G_BURN_1D

!  ----------------------
!  L I N K E D   L I S T S
!  ----------------------

  TYPE, PUBLIC :: list_block
     TYPE(block), POINTER      :: blocks(:)

     INTEGER                   :: MPI_COMM_ROCBURN, rank
     INTEGER                   :: burn_model, TBL_flag, burn_iter,        &
```

```
                                    burn_cell, total_cell

      REAL(DBL)                  :: To_read, pseudo_time
      REAL(DBL), POINTER         :: Tn(:)            ! Buffer for 1D rocburn

      CHARACTER(LEN=80)          :: mname

      TYPE(G_BURN_1D), POINTER  :: g_1d
   END TYPE list_block
```

The pointer `g-1d` is then used by the subroutines (supplied by 1-D combustion model) (and is also used in *Rocburn2D)* but defined by 1-D combustion model. With this algorithm, *Rocburn2D* does not need to have "specific" information about the global variables of the 1-D combustion model (i.e., `g-1d` with derived type `TYPE G_BRUN-1D` defined by individual 1-D combustion mode.). The F90 compiler will point the pointer to the appropriate global variables defined by the selected 1-D combustion model. However, a buffer with certain size is needed (e.g. `INTEGER :: buf(4096)`) for this purpose. The exact mechanism on how this implementation work is unknown at this time. Therefore, there is no formula exist for calculating the size of this buffer. It should be noted that the size of the buffer `buf(:)` used in current implementation is 4096. Currently, this size is determined by trial and error (e.g. gradually increases the size until the code works). The size of `buf(:)` needs to be large enough so that *Rocburn2D* can communicate with the selected 1-D combustion model correctly. When new combustion model is added to *Rocburn2D*, the size of this buffer might need to be changed.

4 Currently Available One-Dimensional Combustion Models

4.1 Brief description of combustion models

There are three combustion models available: *Rocburn_APN*, *Rocburn_PY*, and *Rocburn_ZN*. User should refer to references 3 and 4 for detailed information related to *Rocburn_PY* and references 5 and 6 for *Rocburn_ZN*. A brief description of these models is summarized as:

Rocburn_APN

Characteristics	quasi-steady empirical model (Vieille's or Saint Robert's law $\dot{r}_b = aP^n$)
Condensed phase energy equation	N/A
Condensed phase reaction model	N/A
Gas phase reaction model	N/A
Ignition model	N/A

Source code directory	genx/Rocbun/Codes/Rocburn_APN
I/O directory	genx/Test/RocburnAPY

Rocburn_PY

Characteristics	QSHOD
Condensed phase energy equation	unsteady 1-D heat conduction
Condensed phase reaction model	surface pyrolysis
Gas phase reaction model	quasi-steady large activation energy associated with empirical burning rate law $\dot{r}_b = aP^n$ (see references 3 and 4 for detail)
Ignition model	ignition by convective heat flux implemented (inert heating until $T_s \geq T_{ign}$)
Source code directory	genx/Rocbun/Codes/Rocburn_PY
I/O directory	genx/Test/RocburnPY

Rocburn_ZN

Characteristics	QSHOD (Zeldovich-Novozhilov phenomenological model)
Condensed phase energy equation	unsteady 1-D heat conduction
Condensed phase reaction model	large activation energy surface reaction (Ibiricu and Williams model [7])
Gas phase reaction model	quasi-steady zero activation energy for homogeneous propellant (WSB model [8, 9]), empirical burning law for composite propellant
Ignition model	radiant ignition model under development
Source code directory	genx/Rocbun/Codes/Rocburn_ZN
I/O directory	genx/Test/RocburnZN

where OSHOD is the acronym of Quasi-Steady condensed-phase reaction zone and gas-phase, Homogeneous propellant, One-Dimensional heat feedback, T_s is the propellant surface temperature, and T_{ign} is the specified ignition temperature. For detailed information about the input data and combustion model, please refer to individual combustion model's user guide and developer guide.

4.2 Select combustion model for simulation

To select the combustion model used in the simulation, specify the module_name for combustion model in *GEN3* control file RocstarControl.txt in directory `genx/Test` (using "Test" as an example) as:

1-D combustion model selected	**module_name**
Rocburn_APN	`RocburnAPN`
Rocburn_PY	`RocburnPY`
Rocburn_ZN	`RocburnZN`

The input data file for specifying propellant properties (and will be read by the system code) is in:

1-D combustion model selected	**Input data file**
Rocburn_APN	`genx/Codes/Rocburn/Rocburn_APN/RocburnAPNControl.dat`
Rocburn_PY	`genx/Codes/Rocburn/Rocburn_PY/RocburnPYControl.dat`
Rocburn_ZN	`genx/Codes/Rocburn/Rocburn_ZN/RocburnZNControl.dat`

5 Input and Output (User Interface)

There is no input file required for *Rocburn2D*. Data needed for *Rocburn2D* related to the geometry of the burning surface (i.e. 2-D level) are inherited from the ignitable panes in fluids and are obtained from *Roccom* by calling:

```
!
!   Create interface data in Roccom and allocate memory for them
!
CALL COM_new_window( ioWin)
!   Use the subset of fluid or solid mesh.!   It must use ghost nodes/cells as
!   well in order to visualize.
    CALL COM_clone_attribute( ioWin//".mesh", inSurf//".mesh")
    CALL COM_clone_attribute( ioWin//'.bflag', inSurf//'.bflag')
```

Input data files for propellant specification are managed by 1-D combustion module as mentioned in Section 4.2.

Rocburn2D provides screen dump for the number (and percentage) of cells ignited as follows to monitor the progress of ignition simulation:

```
write(*,*)'ROCBURN iter :: ',G_b%burn_iter,'CELLSIGNITED',              &
               G_b%burn_cell,'PERCENT',                                 &
               dble(G_b%burn_cell)/dble(G_b%total_cell)*100.0d0,G_b%total_cell
```

Output data files (interface data for visulization) and data required for restart, predictor-corrector iterations, and system level sub-cycling are handled by *Rocman*. **All the data communicating through *Roccom* should be in MKS units.** A summary of interface data that needed to communicate with *Rocman* is listed as:

Data in incoming buffers

```
!
!   Incoming data
!
    CALL COM_new_attribute( ioWin//".pf_alp", 'e', COM_DOUBLE, 1, "Pa")
    CALL COM_inherit_attribute( ioWin//".bflag", fluidWin//".bflag")

    IF ( .NOT. is_APN) THEN
       CALL COM_new_attribute( ioWin//".centers", 'e', COM_DOUBLE, 3, "m")
       CALL COM_new_attribute( ioWin//".qr_alp", 'e', COM_DOUBLE, 1, "W/m^2")
       CALL COM_new_attribute( ioWin//".qc_alp", 'e', COM_DOUBLE, 1, "W/m^2")
       CALL COM_new_attribute( ioWin//".rhos_alp", 'e', COM_DOUBLE, 1, "kg/m^3")
       CALL COM_new_attribute( ioWin//".Tf_alp", 'e', COM_DOUBLE, 1, "K")
!!!    CALL COM_new_attribute( ioWin//".To_alp", 'e', COM_DOUBLE, 1, "K")
       ! Reuse the bflag created in fluids
       CALL COM_resize_array(ioWin//".centers")
       CALL COM_resize_array(ioWin//".qr_alp")
       CALL COM_resize_array(ioWin//".qc_alp")
       CALL COM_resize_array(ioWin//".rhos_alp")
       CALL COM_resize_array(ioWin//".Tf_alp")

    END IF
```

Data in outgoing buffers

```
!
!   Outgoing data
!
    CALL COM_new_attribute( ioWin//".rb", 'e', COM_DOUBLE, 1, "m/s")
    CALL COM_new_attribute( ioWin//".Tflm", 'e', COM_DOUBLE, 1, "K")
    CALL COM_resize_array(ioWin//".rb")
    CALL COM_resize_array(ioWin//".Tflm")

    CALL COM_window_init_done( ioWin)
```

Data for interpolation if sub-cycling for individual cells are needed

```
IF ( .NOT. is_APN) THEN
       CALL COM_clone_attribute( intWin//".pf_old", ioWin//".pf_alp")
       CALL COM_clone_attribute( intWin//".qc_old", ioWin//".qc_alp")
       CALL COM_clone_attribute( intWin//".qr_old", ioWin//".qr_alp")
       CALL COM_clone_attribute( intWin//".rhos_old", ioWin//".rhos_alp")
       CALL COM_clone_attribute( intWin//".Tf_old", ioWin//".Tf_alp")
!!!    CALL COM_clone_attribute( intWin//".To_old",  ioWin//".To_alp")
    END IF
```

Data for profile history

```
IF ( .NOT. is_APN) THEN
       CALL COM_clone_attribute( intWin//".Toa", ioWin//".Tflm")
       IF ( comp_filmcoeff) THEN
          CALL COM_new_attribute( intWin//".dist", 'e', COM_DOUBLE, 1, "m")
          CALL COM_resize_array(intWin//".dist")
       ENDIF
       CALL COM_new_attribute( intWin//".temp", 'e', COM_DOUBLE, nxmax, "K")
       CALL COM_new_attribute( intWin//".fr", 'e', COM_DOUBLE, 1, "")
       CALL COM_resize_array(intWin//".temp")
       CALL COM_resize_array(intWin//".fr")
END IF
```

6 Building and Running

The source codes of *Rcobun2D* are located in directory `genx/Codes/Rocburn` and the source codes for 1-D combustion models are located in directories `genx/Codes/Rocburn/Rocburn_APN`, `genx/Codes/Rocburn/Rocburn_PN`, and `genx/Codes/Rocburn/Rocburn_ZN`. To compile *Rocburn2D* and 1-Dcombustion models, use the Makefile within these directories. The Makefile and Makfile.baskic for *Rcoburn2D* are listed as:

Makefile

```
# Makefile file Rocburn
srcdir          = .
top_srcdir      = ..
PREFIX          = $(top_srcdir)
include $(srcdir)/Makefile.basic
```

Makefile.basic

```
 # Makefile file Rocburn


LIBDIR          = $(PREFIX)/lib
```

```
BINDIR          = $(PREFIX)/bin

LIBSUF          = so
LIBBURN         = $(LIBDIR)/libRocburn.$(LIBSUF)

vpath   %.f90   $(srcdir)
vpath   %.so    $(LIBDIR)

LIBS_1D         = Rocburn_APN/libRocburn_APN.a \
                  Rocburn_PY/libRocburn_PY.a \
                  Rocburn_ZN/libRocburn_ZN.a

OBJS1           = 01.burn_interface_data.o
OBJS2           = calcdist.o rocburn_2D.o  Rocburn_load_module.o
OBJS            = $(OBJS1) $(OBJS2)

#include the common makefile components
COMHOME         = $(top_srcdir)/Roccom
BUILD_COMHOME           = ../Roccom
include $(COMHOME)/Makefile.common

all : $(LIBBURN)



$(LIBBURN):     $(OBJS) $(LIBS_1D)

#--------------------------------- dependencies
.PHONY:         FORCE
$(OBJS2) : $(OBJS1)

# Disabled debugging info for rocburn_2D.o to overcome compiler bug on tungsten
rocburn_2D.o : OPTS_G=

rocburn_2D.o : calcdist.o $(COMHOME)/include/roccomf90.h

Rocburn_load_module.o: rocburn_2D.o $(LIBS_1D)

$(LIBS_1D) : FORCE
        @$(MAKE) -C $(@D) $(@F)
        -$(LN) */*.mod .


#--------------------------------- clean
clean:
        $(RM) *.o *.$(MX) *.int *.bif *.s $(LIBBURN) a.out core
        for d in $(dir $(LIBS_1D)) ; do $(MAKE) -C $$d clean; done
```

The executable generated by this make file is a library `libRocburn.so` stored in directory `genx/lib` that includes all the executables (`libRocburn_APN.a`, `libRocburn_PY.a`, and `libRocburn_ZN.a`) for 1-D combustion models. The executable for 1-D combustion model can be obtained by compiling the source codes using the Makefile within its own subdirectory. The sample input data file for individual combustion model can also be found there. But these input data files should be copied to the working directory during run time. The executable for the integrated system code is located in `genx/Codes` as genx.x.

The combustion model used in system simulation can be determined at run time by specifying the module_name for combustion mode in the control file GENXControl.txt and providing corresponding input data file for propellant properties (see Section 4.2 for detail). After providing input data file for the combustion model selected, please refer to *GEN3* User's Guide [1] for running the simulation using *GEN3* code.

7 Adding New Combustion Model

Adding new combustion model into *Rocbrun* can be completed by adding the **bold** part of the shown example listed below into the Makfile.basic and Rocburn_load_module.f90 in directory `gen2_5/Solver/Rocburn` (use *Rocburn_NEW* as an example):

Makefile.basic

```
•
•
•
LIBBURN         = $(LIBDIR)/libRocburn.$(LIBSUF)

vpath   %.so    $(LIBDIR)

LIBS_1D         = Rocburn_APN/libRocburn_APN.a \
                  Rocburn_NEW/libRocburn_NEW.a
                  Rocburn_PY/libRocburn_PY.a \
                  Rocburn_ZN/libRocburn_ZN.a

•
•
•
```

Rocburn_load_module.f90

```
•
•
•
SUBROUTINE ROCBURN_INIT_FUNCS_APN( mname)
    USE M_ROCBURN_1D_APN
    CHARACTER(*), INTENT(IN) :: mname

    CALL COM_set_external( mname//".init_0d", 0, INITIALIZE_0D)
    CALL COM_set_external( mname//".init_1d", 0, INITIALIZE_1D)
    CALL COM_set_external( mname//".finalize_0d", 0, FINALIZE_0D)
    CALL COM_set_external( mname//".get_burn_rate", 0, GET_BURNING_RATE_1D)

  END SUBROUTINE ROCBURN_INIT_FUNCS_APN

SUBROUTINE ROCBURN_INIT_FUNCS_NEW( mname)
    USE M_ROCBURN_1D_NEW
    CHARACTER(*), INTENT(IN) :: mname

    CALL COM_set_external( mname//".init_0d", 0, INITIALIZE_0D)
    CALL COM_set_external( mname//".init_1d", 0, INITIALIZE_1D)
```

```
    CALL COM_set_external( mname//".finalize_0d", 0, FINALIZE_0D)
    CALL COM_set_external( mname//".get_film_coeff", 0, COM_NULL)
    CALL COM_set_external( mname//".get_time_step", 0, COM_NULL)
    CALL COM_set_external( mname//".get_burn_rate", 0, &
                                     GET_BURNING_RATE_1D)

  END SUBROUTINE ROCBURN_INIT_FUNCS_NEW


  SUBROUTINE ROCBURN_INIT_FUNCS_PY( mname)
    USE M_ROCBURN_1D_PY
    CHARACTER(*), INTENT(IN) :: mname

    CALL COM_set_external( mname//".init_0d", 0, INITIALIZE_0D)
    CALL COM_set_external( mname//".init_1d", 0, INITIALIZE_1D)
    CALL COM_set_external( mname//".finalize_0d", 0, FINALIZE_0D)
    CALL COM_set_external( mname//".get_film_coeff", 0, GET_FILM_COEFF_1D)
    CALL COM_set_external( mname//".get_time_step", 0, GET_TIME_STEP_1D)
    CALL COM_set_external( mname//".get_burn_rate", 0, GET_BURNING_RATE_1D)

  END SUBROUTINE ROCBURN_INIT_FUNCS_PY



  SUBROUTINE ROCBURN_INIT_FUNCS_ZN( mname)
    USE M_ROCBURN_1D_ZN
    CHARACTER(*), INTENT(IN) :: mname

    CALL COM_set_external( mname//".init_0d", 0, INITIALIZE_0D)
    CALL COM_set_external( mname//".init_1d", 0, INITIALIZE_1D)
    CALL COM_set_external( mname//".finalize_0d", 0, FINALIZE_0D)
    CALL COM_set_external( mname//".get_film_coeff", 0, GET_FILM_COEFF_1D)
    CALL COM_set_external( mname//".get_time_step", 0, GET_TIME_STEP_1D)
    CALL COM_set_external( mname//".get_burn_rate", 0, GET_BURNING_RATE_1D)

  END SUBROUTINE ROCBURN_INIT_FUNCS_ZN

•
•
•
  G_b%mname = module_name
  G_b%TBL_flag = NO_TBL
  IF ( module_name == "RocburnAPN") THEN
     G_b%burn_model = MODEL_APN
  ELSE IF ( module_name == "RocburnNEW") THEN
     G_b%burn_model = MODEL_NEW
  ELSE IF ( module_name == "RocburnPY") THEN
     G_b%burn_model = MODEL_PY
  ELSE IF ( module_name == "RocburnZN") THEN
     G_b%burn_model = MODEL_ZN
  ELSE
     PRINT *, "Rocburn-2D: Unknown module name", module_name
     PRINT *, "Rocburn-2D: Use APN instead", module_name
     G_b%burn_model = MODEL_APN
  END IF
•
•
•
!!! Now initialize the 1D module
  IF ( G_b%burn_model == MODEL_PY) THEN
     CALL ROCBURN_INIT_FUNCS_PY( module_name)
   ELSE IF ( G_b%burn_model == MODEL_NEW) THEN
     CALL ROCBURN_INIT_FUNCS_NEW( module_name)
```

```
  ELSE IF ( G_b%burn_model == MODEL_ZN) THEN
     CALL ROCBURN_INIT_FUNCS_ZN( module_name)
  ELSE
     CALL ROCBURN_INIT_FUNCS_APN( module_name)
  END IF
•
•
•
```

Hopefully these are the only changes needed in *Rocburn2D* for adding a new combustion model. It should be noted that it might become necessary to change the size of buffer used in the placeholder for 1-D global variables (see Sec. 3.3). The new combustion model *Rocburn_NEW* needs to complete the following suggested tasks (using the examples mentioned above):

- Create a subdirectory `Rocburn_NEW` under `genx/Codes/Rocburn` to store the source codes and input data files
- Define the `TYPE G_BURN_1D` for the global variables used in the combustion model to be passed between difference subroutines
- Create a module `M_ROCBURN_1D_NEW` to provide the mandatory subroutines specified in Section 3.2

8 Test Problem

The firing of NAWC Motor #13 is used as a test problem. Experimental measurement of head end pressure is available for comparison. The NAWC Motor #13 is a 33" long, 4.8" in diameter tactical motor [10]. This is a cylindrical, center perforate grain (3.0" in diameter) with a slight taper at the aft end, as shown in Figure 2. The nozzle throat diameter of this motor is 2.08". The solid propellant used (NWR11b) in the experimental test has burning rate of 0.154 in/sec at 500.0 psi and pressure exponent n = 0.461. Properties of the AP composite solid propellant NWR11b, including both reported measurements and assumed modeling parameter values, can be found in the input data files `Rocburn_APN_0d.dat` for *Rocburn_APN* (see Table 1), `input_py.dat` for *Rocburn_PY* (see Table 2), and `Rocburn_ZN_0d.dat` for *Rocburn_ZN* (see Table 3).

Since *Rocburn_APN* uses the empirical burning rate law $\dot{r}_b = aP^n$, the input date required to model the burning rate are the values of *a* and *n*. The flame temperature and initial propellant temperature are also needed (see Table 1). The flame temperature can be obtained from direct measurement or from numerical calculation. The units of these parameters can be found in the data file. Additional thermo-physical properties are needed for *Rocburn_PY* and *Rocburn_ZN* model (see Tables 2 and 3).

It should be noted that *Rocburn_PY* uses the empirical burning rate mode in the form of $\dot{r}_b = a(P/P_{ref})^n$. This form includes a reference pressure P_{ref} in the equation. User should calculate the value of *a* accordingly. Other parameters are related to the *Rocbrun_PY* combustion model (`Ac, eg_ru, ec_ru, alfac, C, lamg, delt, Tstar0, To`) and ignition model (`Tignition, Tsurf, film_cons, ixsymm, and x_surf_burn`). Brief description of these parameters can be found in the input data file `input_py.dat` (Table 2). See Rocburn_PY User's Guide [3] for detailed description.

Table 3 lists the parameters required for *Rocburn_ZN* combustion model. For homogeneous propellant, the combustion model number 1 (i.e. ZN_WSB) should be selected. The values of *a* and *n* ($\dot{r}_b = aP^n$) in the data file are not used for this model. The WSB combustion model [8, 9] is used in this case. All other parameters are either measured or modeled for a specific propellant. Usually these parameters need to be modified for different propellant in order to achieve better agreement in predicted burning rate, burning rate pressure exponent, pressure and/or radiation response, propellant surface temperature, temperature sensitivity, and so on. The combustion model number 2 (i.e. ZN_Empirical) should be used for composite propellant associated with *a* and *n* supplied. An empirical gas phase combustion model is used. The parameters related to gas phase combustion (`Bg` and `lamg`) are not used. Similar to homogeneous propellant, other parameters are also specific to a particular propellant. Adjustment of these parameters for other composite propellant is often necessary.

Results for NAWC Motor #13 simulation are shown in Figs. 3 and 4. The head end pressures calculated by *Rocbrun_APN* and *Rocburn_ZN* without ignition simulation (e.g. assuming all the propellant ignited from onset) are shown in Figure 3 (note: these are the old results from *GEN1* but is shown here to indicate what to be expected, to be replaced when *GEN3* result is available). *Rocburn_PY* result with ignition simulation is shown in Figure 4.

9 References

1. *GEN3* User's Guide

2. *GNE3* Developers' Guide

3. *Rocburn_PY* User's Guide

4. *Rocburn_PY* Developer's Guide

5. *Rocburn_ZN* User's Guide

6. *Rocburn_ZN* Developer's Guide

7. Ibiricu, M. M., and Williams, F. A. (1975), "Influence of Externally Applied Thermal Radiation on the Burning Rates of Homogeneous Solid Propellants," *Combustion and Flame*, Vol. 24, pp. 185-198.

8. Brewster, M. Q., Ward, M. J., and Son, S. F. (2000a), "Simplified Combustion Modeling of Double Base Propellant: Gas Phase Chain Reaction Vs. Thermal Decomposition," *Combustion Science and Technology*, Vol. 154, pp. 1-30.

9. Ward, M. J., Son, S. F., and Brewster, M. Q. (1998a), "Steady Deflagration of HMX with Simple Kinetics: A Gas Phase Chain Reaction Model," *Combustion and Flame*, Vol. 114, pp. 556-568.

10. Blomshield, F. S., J. E. Crump, H. B. Mathes, and M. W. Beckstead, "Stability Testing and Pulsing of Full Scale Tactical Motors," NAWCWPNS TP 8060, 1996.

Table 1. Input data file `RocburnAPNControl.dat` for NWR11b solid propellant in *Rocburn_APN*

```
0.07696    a in rb=a*P^n,  rb in cm/sec and P in atm, a_p (cm/sec)
0.461      n in rb=a*P^n,  rb in cm/sec and P in atm, n_p
1          Maximum_number_of_spatial_nodes,_nxmax
2855.0     adiabatic flame temperature, Tf_adiabatic (K)
298.00     initial temperature        , To_read      (K)
Rocburn_2D_Output/Rocburn_APN

Solid Propellant Properties for NAWC Motor #13 ONLY
```

Table 2. Input data file `RocburnPYControl dat` for NWR11b solid propellant in *Rocburn_PY*

```
.391            a_p             IN rb = a_p*(P/Pref)^n,  rb in cm/sec and P in atm
0.46100         n_p             IN rb = a_p*(P/Pref)^n,  rb in cm/sec and P in atm
34              Pref            IN rb = a_p*(P/Pref)^n,  atm
180000.0        Ac              Condensed_phase_prefactor,(cm/s)
24000.0         eg_ru           Gas_phase_activation_temperature,_eg_ru_(K)
12500.0         ec_ru           Condensed_phase_activation_temperature,_[ec_ru]_(K)
1.72e-3         alfac           Condensed_phase_thermal_diffusivity,_alfac_(cm^2/s)
0.350           C               Specific_heat_(gas_and_condensed_phases),_C_(cal/g-K)
2.00e-4         lamg            Gas_phase_thermal_conductivity,_lamg_(cal/cm-s-K)
3.00E-6         delt            Time_step,_delt_(s);_WATCH_STABILITY
2               igrid           Grid_control_distribution,_igrid;_1=EXP;_2=BL
100             numx            number of points in propellant depth
-0.200          xmax            Maximum_x_location,_xmax_(m);_MUST_BE_NEGATIVE
1.010           beta            Grid_stretching_parameter,_beta
2855.0          Tstar0          adiabatic flame temperature, Tstar0    (K)
300.00          To              cold temperature,            To        (K)
835.00          Tignition       ignition temperature,        Tignition (K)
300.00          Tsurf           surface  temperature,        Tsurf     (K)
560.08          film_cons       constant in film coefficient [ W/ (m^2 K) ]
0               ixsymm          axisymmetric initial burning, use x_surf_burn
0.18000         x_surf_burn     last surface x location burning from the onset
1.00000E+8      press_max       maximum pressure allowed to be passed in [Pa]
100.00          press_min       minimum pressure allowed to be passed in [Pa]
100.0000        rb_max          maximum burn rate allowed [m/sec]
-1.00000E-9     rb_min          minimum burn rate allowed [m/sec]
10000.          Tf_max          maximum gas temperature allowed [K]
290.00          Tf_min          minimum gas temperature allowed [K]
0               TABUSE          1 USE a table algorithm, 0 USE analytical results
RBRNtable.dat   TABNAME         name of the file w/ table
```

Table 3. Input data file `RocburnZNControl.dat` for NWR11b solid propellant in *Rocburn_ZN*

```
2          Combustion model (1= ZN_WSB, 2=ZN_empirical, 3=a*P^n), Model_combustion
0.07696    a in rb=a*P^n,  rb in cm/sec and P in atm, a_p (cm/sec)
0.461      n in rb=a*P^n,  rb in cm/sec and P in atm, n_p
8.00e10    Condensed_phase_prefactor,_Ac_(1/s)
1.66e-3    Gas_phase_prefactor,_Bg_(cal^2/cm^3-atm^2-g-s-K^2)
29000.0    Condensed_phase_activation_energy,_Ec_(cal/mol)
0.00       Condensed_phase_chemical_release,_Qc_(cal/g)
700.0      Gas_phase_chemical_heat_release,_Qg_(cal/g)
1.00e-3    Condensed_phase_thermal_diffusivity,_alfac_(cm^2/s)
0.350      Specific_heat_(gas_and_condensed_phases),_C_(cal/g-K)
1.7026     Condensed_phase_density,_rhoc_(g/cm^3)
2.00e-4    Gas_phase_thermal_conductivity,_lamg_(cal/cm-s-K)
```

```
25.49      Gas_phase_molecular_weight,_MW_(g/mol)
100.0      Condensed_phase_absorption_coefficient,_Ka_(1/cm)
101        Maximum_number_of_spatial_nodes,_nxmax
101        Number_of_spatial_nodes,_nx
1.0e-6     Time_step,_delt_(s);_WATCH_STABILITY
2          Grid_control_distribution,_igrid;_1=exponential;_2=boundary_layer
-0.200     Maximum_x_location,_xmax_(cm);_MUST_BE_NEGATIVE
1.010      Grid_stretching_parameter,_beta
1.0e-4     Surface_temperature_tolerance_for_convergence,_tol_Ts(K)
200        Maximum_#_of_iterations,_itermax
2855.0     adiabatic flame temperature, Tf_adiabatic (K)
298.00     initial temperature        , To_read      (K)
0          ignition simulation flag, ign_flag (0= NO, 1= YES)
380.00     cold boundary temperature  , To_cold      (K)
Rocburn_ZN_Out/

Solid Propellant Properties for NAWC Motor #13 ONLY
```

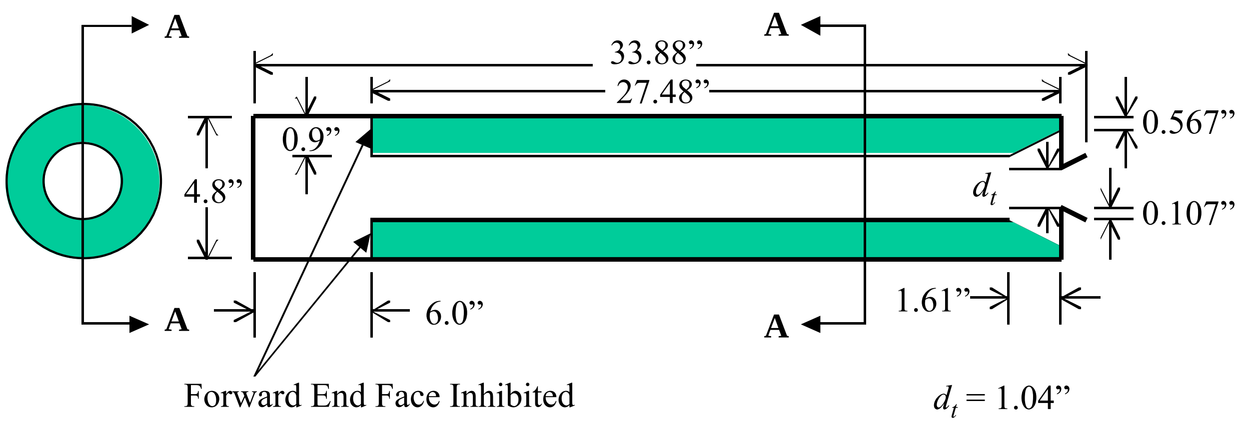

Figure 2. NAWC tactical motor13 geometry.

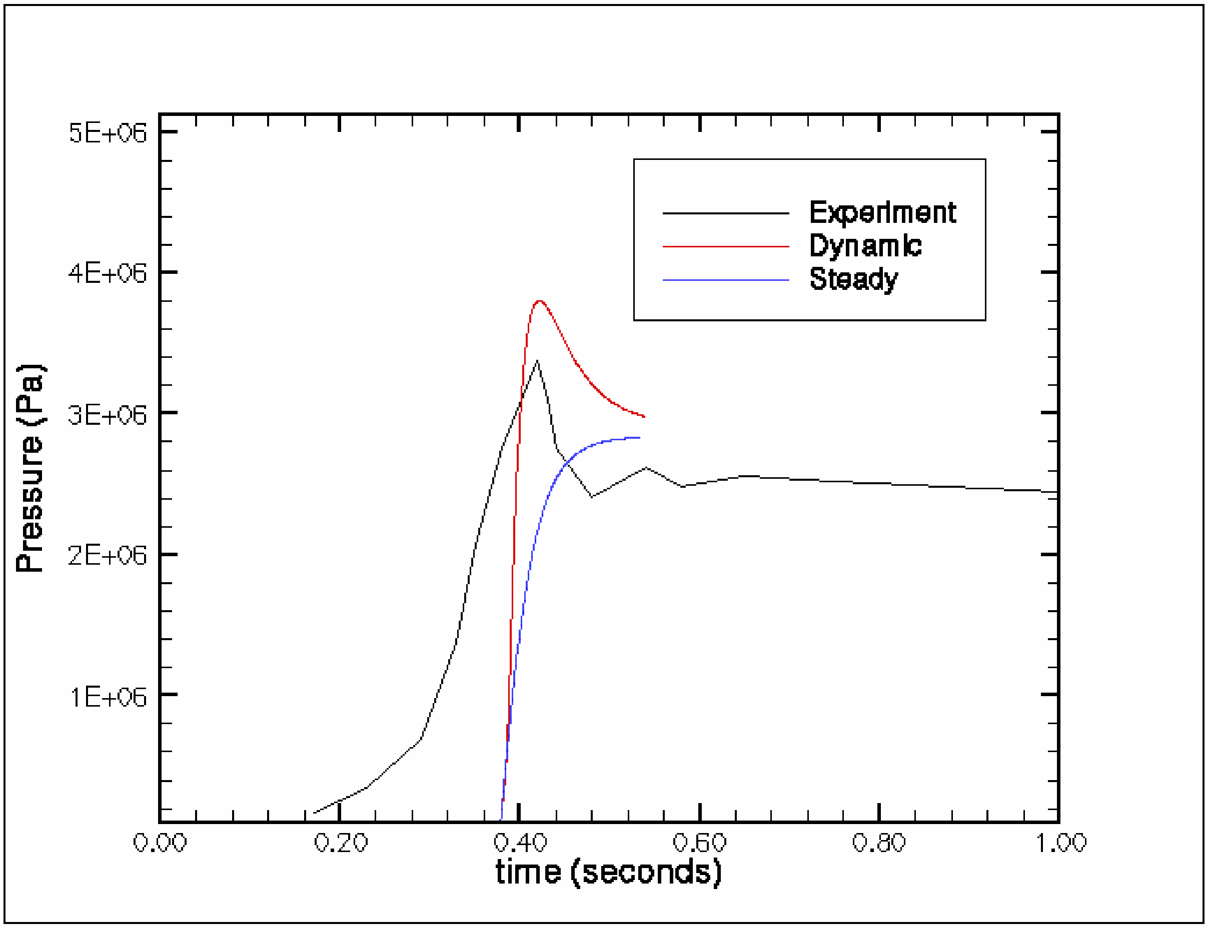

Figure 3. *Rocburn_ZN* results: pressure-time history for NAWC motor #13 without ignition simulation (i.e. assuming entire propellant ignited from onset). Result denoted with "Steady" is obtained from *Rcocbun_APN* and the result denoted with "Dynamic" is from *Rocburn_ZN*.

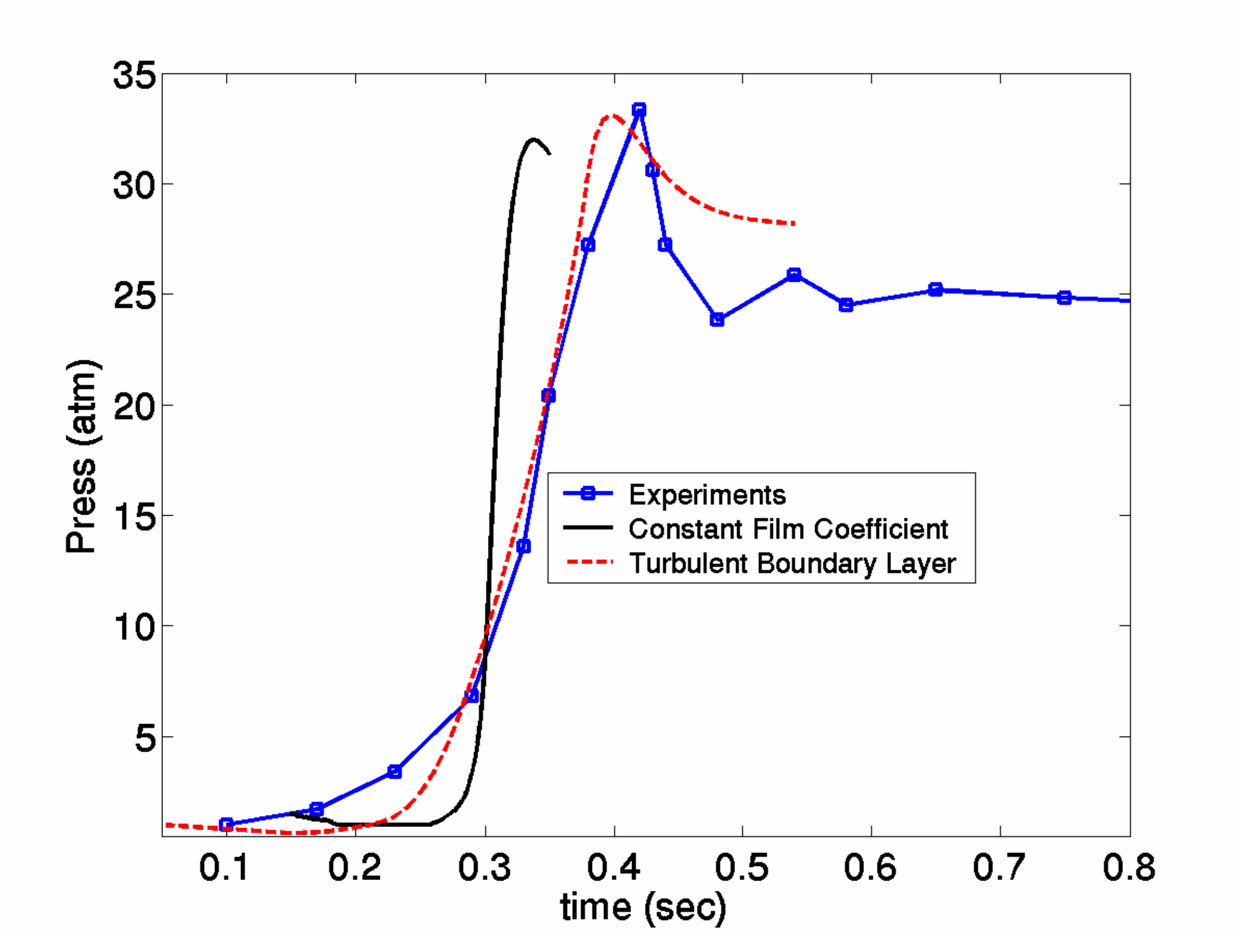

Figure 4. *Rocburn_PY* results: pressure-time history for NAWC motor #13 with ignition simulation.