Gpyro – A Generalized Pyrolysis Model for Combustible Solids
Users' Guide

Version 0.8154 – SVN 0489
May 19, 2016

THIS DOCUMENT LAGS THE CURRENT GPYRO VERSION IN THE REPOSITORY AND IS IN THE PROCESS OF BEING UPDATED.

Chris Lautenberger

Reax Engineering Inc.
1921 University Ave
Berkeley, CA 94704

510-629-4930 x801
lautenberger@reaxengineering.com

Acknowledgments

The financial support of NSF and NASA is gratefully acknowledged. Gpyro development was funded as part of NSF Award 0730556, "Tackling CFD Modeling of Flame Spread on Practical Solid Combustibles" (September 2007-September 2010). Development of early Gpyro versions was funded by the NASA Graduate Student Researcher Program under Grant #NNC-04HA08H, "Piloted Ignition and Flame Spread on Composite Materials in Partial and Normal Gravity" (August 2004 – August 2007).

Esther Kim at WPI and Amanda Dodd at UC Berkeley have played important roles in model development by serving as beta testers and exercising Gpyro in ways that were not contemplated during its initial development. I would like to thank them for their patience, their bug reports, and their suggestions as to how to improve Gpyro. Bryan Klein at Thunderhead Engineering helped get the source code and other files under Subversion control on Google Code (before the repository was moved to a Reax Engineering server). Thomas Steinhaus from the University of Edinburgh showed great patience as we worked together to get Gpyro running on a cluster there.

Others that have contributed to the development of Gpyro over the years include Nicolas Bal, Bangalore Yashwanth, Lucie Hasalova, Davood Zeinali, Xinyan Huang, Salah Benkorichi, and probably many others that I am forgetting!

Contents

1.0 INTRODUCTION

This document is the Users' Guide to Gpyro, a generalized pyrolysis model for combustible solids (see http://reaxengineering.com/trac/gpyro). Technical details can be found in the Gpyro Technical Reference [1], the PhD dissertation on which Gpyro is based [2], and several journal and conference publications [3-9]. Both the Users' Guide and Technical Reference will be continuously updated as Gpyro evolves.

1.1 What is Gpyro?

Gpyro is open-source, freely available computer software that simulates thermo-chemical processes occurring in thermally stimulated solids (loosely, it is a pyrolysis model). It can be used for 0D (lumped), 1D, 2D, or 3D simulations of thermally stimulated solids, including thermal and thermo-oxidative decomposition of condensed-phase species. Gpyro is coupled to Version 6.2.0 of NIST's Fire Dynamics Simulator (FDS)[1] [10] where it can be applied as a boundary condition to facilitate coupled modeling of ignition and flame spread. That is, calculations can be conducted either as standalone simulations or in a coupled simulation where Gpyro is applied as a boundary condition in FDS.

An integral part of Gpyro is a material property estimation program that can be used to estimate material properties from laboratory tests such as ThermoGravimetric Analysis (TGA), Differential Scanning Calorimetry (DSC), and conventional flammability tests such as the Cone Calorimeter and the Fire Propagation Apparatus (FPA). The basic property estimation routine was originally limited to genetic algorithm optimization [7] and was later extended to additional methods [8]. The property estimation routines use Message Passing Interface (MPI) to reduce run times with parallel processing. The property estimation routines are "embarrassingly parallel", and Gpyro's property estimation routines have been successfully run on 128+ computational cores with near-linear speedup.

Since Gpyro is generalized, it can be configured to solve the same equations as are solved by the FDS pyrolysis model (by setting `FDSMODE = .TRUE.` on the `&GPYRO_GENERAL` line). This makes it possible to run Gpyro in "FDS mode" to estimate material properties for use in off the shelf versions of FDS. Gpyro can write both FDS5 and FDS6 formatted `&SURF`, `&MATL`, and `&RAMP` namelist groups lines that can be copied and pasted into FDS input files. It is the user's responsibility to confirm that the Gpyro calculations match the FDS calculations for the particular case under consideration.

Gpyro is a research code under active development and it is constantly evolving. The user is advised to frequently check http://reaxengineering.com/trac/gpyro for updates, bug fixes, improved documentation, sample input files, etc.

[1] Gpyro and its coupling to FDS is not supported or endorsed by NIST or VTT in any way.

1.2 Is Gpyro Verified? Validated? Has Gpyro been published?

These questions are asked frequently. Inevitably, semantics come into play when discussing model verification and validation (V+V), and different people and organizations use the terms to mean different things. Rather than getting bogged down in semantics, the ways in which Gpyro has been exercised are summarized below; the user can decide if this constitutes V+V.

Gpyro has been shown to accurately reproduce several exact solutions with relevance to pyrolysis (see Section 3.4 of Ref. [2] and more recent discussions of the extension of Gpyro to 3D [9]). Additional comparisons of model calculations and exact solutions have been conducted as part of the model development process and the author is not aware of any instances where the numerical calculations do not match the analytical solutions. Instantaneous and cumulative energy balances have been conducted to confirm that Gpyro globally conserves mass, energy, and species. The effects of timestep, grid spacing, and thermophysical parameters have been determined (see Chapter 4 of Ref. [2]) and are self-consistent. Gpyro has been used to simulate the thermal and thermo-oxidative pyrolysis behavior of several real fuels and the reader is referred to the published literature for details [3-9].

1.3 Philosophy behind Gpyro

Rather than hard-coding a certain level of complexity, Gpyro provides the user with the ability to include as much or as little complexity in a calculation as desired. In its simplest form, Gpyro is a 1D transient heat conduction code for a constant property inert (nonreactive) solid. In its most complicated form, Gpyro can be used to solve 3D conjugate heat transfer, fluid flow/pressure evolution, gaseous species diffusion/convection in a multi-component, heterogeneous, reacting porous medium with temperature-dependent thermophysical properties, in-depth radiation absorption, radiation heat transfer across pores, multi-step heterogeneous decomposition kinetics of arbitrary order or complex reaction model, and multi-step homogeneous gas-phase reactions. The level of complexity included in a simulation, and the requisite number of model input parameters/material properties, is decided entirely by the user.

Gpyro development efforts have focused on providing generalized software containing physical phenomena that can be enabled or disabled as desired by the user. Similarly, the material property estimation routines require the user to specify valid parameter ranges as well as other variables/flags that affect how the material property estimation process occurs. Gpyro, and its associated material property estimation routines, are merely tools that can be used to determine which physics matter under which circumstances. Equally as important, Gpyro can be used to develop “validated” sets of material properties that can be used to model the pyrolysis behavior of solid materials under fire conditions.

1.4 Types of problems that can be analyzed with Gpyro

Much effort has been devoted to keeping Gpyro as general as possible to expand the types of problems that can be analyzed. The following is a brief partial list of the types of problems that Gpyro has been used to simulate at some point during its development:

- TGA experiments (primarily for estimating reaction kinetics)
- DSC experiments (primarily for estimating heats of reaction)
- Cone Calorimeter and FPA experiments (primarily for estimating reaction kinetics, heats of reaction, and thermophysical properties)
- Thermal pyrolysis and surface regression of noncharring thermoplastics
- Thermal pyrolysis of charring solids
- Thermal pyrolysis of heterogeneous fiber reinforced polymer composites
- Heating, gasification, and swelling of intumescent coatings, and the thermal resistance offered by such coatings
- Effect of ambient oxygen concentration on thermo-oxidative pyrolysis of charring solids
- Smolder wave propagation in porous media
- Transition from smolder to flame in porous media
- Propellant cookoff in a confined housing
- Self-heating and spontaneous ignition of rags soaked in linseed oil
- Ignition of fuel beds by embers, firebrands, and sparks
- Laminar opposed flow flame spread (coupled to FDS)

Other problems that could be simulated with Gpyro either "out of the box" or with minor modifications include:

- Concrete spalling
- Ablation of sacrificial protective coatings
- Behavior of thermally stimulated anisotropic solids (wood, fiber reinforced polymer composites)
- Biomass pyrolysis in fast/flash pyrolysis reactors

1.6 How to use this guide

The best way to use this guide is not necessarily to read it cover to cover. It is recommended that new users run the sample cases included with the Gpyro distribution and then read the sections of this guide as necessary to understand these samples.

1.7 Version control and bug reports

Gpyro and its associated documentation are continuously evolving. The source code, documentation, sample files, and build scripts/makefiles are under Subversion control and a Trac environment is set up at http://reaxengineering.com/trac/gpyro. Bug reports, feature requests, questions, etc. can be entered into Trac's ticketing system at the above URL (you will have to send an email to Chris Lautenberger with a username and password for the Trac system before you can create a ticket). User feedback is a critical part of the update and improvement process, so feedback is welcome and encouraged. Would new physics, features, or capabilities be useful? Is something not clear in the Users' Guide or Technical Reference? Is something not working as you would expect? Does Gpyro give a runtime error?

2.0 GETTING STARTED

Gpyro is a console application that is executed from a Windows command prompt or Linux console. This chapter describes how to download, install, run, and compile Gpyro under Windows and Linux.

2.1 Downloading Gpyro

There are two ways to download Gpyro. The first way is via Subversion checkout, and the second is via download links on Gpyro's Trac page (http://reaxengineering.com/trac/gpyro).

As incremental changes are made to Gpyro's source files, documentation, or sample input files, they are committed to the Subversion (SVN) repository. Subversion tracks all changes, making it possible to dial back to any previous version of Gpyro. Since commits are often made every few days, the Subversion repository contains the "bleeding edge" files. If you would like to work with the most recent files, Subversion is the way to go. A working copy of Gpyro can be checked out either from the command line or using a SVN Graphical User Interface. Instructions for SVN checkout can be found on the Trac page above.

In addition to SVN checkout, Gpyro can be downloaded via links at http://reaxengineering.com/trac/gpyro. For example, Gpyro version 0.8149 can be downloaded from http://reaxengineering.com/media/gpyro/gpyro_0.8149.zip. Gpyro is packaged in a .zip archive named gpyro_x.yyyy.zip where x.yyyy indicates the Gpyro version number (for the above URL, x.yyyy is 0.8149). These zip archives contain precompiled Windows executable files, Microsoft Visual Studio .net Solution files, Linux Makefiles and a build script, sample input files, and documentation. Due to portability issues, Linux executables are not included. Whereas the Subversion repository is updated every few days, the download links are updated every few weeks to every few months, so they don't necessarily contain the most recent files (which can be obtained by Subversion checkout).

2.2 Installation – Windows

There no Windows installer for Gpyro. The Windows "installation" process involves copying the appropriate .exe files to a directory that is included in your PATH (Google "set PATH Windows" for help) so that Windows can find the executable files. The standalone Gpyro executable file is named `gpyro.exe` (or gpyro_openmp.exe for the multithreaded version), the Gpyro property estimation program is named `gpyro_propest.exe`, and the coupled FDS/Gpyro executable is `fds6_mpi_gpyro.exe`. These executable files are located in the relative directory build\win and in the .zip archive and Subversion repository. If you don't want to modify your PATH, then you can copy the .exe files to each directory from which you will run Gpyro. To avoid confusion, you may find it useful to rename the executable files to explicitly indicate their version number, *e.g.* `gpyro_0.8149.exe`.

If you plan to conduct 2D or 3D simulations, then you should also install NIST's Smokeview because Smokeview is used to visualize 2D Gpyro simulations. Smokeview can be downloaded from https://pages.nist.gov/fds-smv/downloads.html.

In order to run `gpyro_propest.exe`, an MPI installation must be correctly installed or a run-time error will result. Gpyro is currently compiled against MS-MPI v7, which can be downloaded from:
https://msdn.microsoft.com/en-us/library/windows/desktop/bb524831(v=vs.85).aspx

2.3 Installation – Linux

There is currently no installer or configure script for installing Gpyro in Linux. Instead, you will have to compile Gpyro using compilers installed on your system and simple build scripts distributed with Gpyro. This has been tested with Intel fortran and Gnu's gfortran but should work with other compilers as well. To compile, cd to the relative directory gpyro/build/linux and execute one of the .sh script files in that directory, *e.g.* "`sh make_gnu.sh`" to run the Gnu gfortran build script. If all goes well, this will build executables and copy them to /usr/local/bin, you can change this behavior in the shell scripts if you desire. Gpyro's property estimation and the coupled Gpyro / FDS executable require MPI to build and run, we use OpenMPI 1.10.2:
https://www.open-mpi.org/software/ompi/v1.10/downloads/openmpi-1.10.2.tar.gz

If you plan to conduct 2D or 3D simulations, then you should also install NIST's Smokeview because Smokeview is used to visualize 2D Gpyro simulations. Smokeview can be downloaded from https://pages.nist.gov/fds-smv/downloads.html.

2.4 Running Gpyro (standalone simulation)

Gpyro is executed from the Windows command prompt or the Linux shell as follows:

```
gpyro input_file.data
```

or

`gpyro_openmp input_file.data` (for the multithreaded version)

where `input_file.data` is a text file that specifies the geometry, boundary conditions, thermophysical properties, reaction kinetics, etc. See Section 3 for an explanation of input files. The format of this input file is described in Section 3.

2.5 Running Gpyro (property estimation program)

The Gpyro property estimation algorithm is a separate executable file from the standalone Gpyro executable. It is run in a way that is similar to the standalone executable, but no input file name is specified:

```
gpyro_propest
```

The reason that no input file is specified is that when running the Gpyro property estimation program, the input file name is assumed to be `gpyro.data`. The filename is hardcoded to simplify running in parallel with Message Passing Interface (MPI). See Section 3 for an explanation of input files. As described in Section 3, when running the property estimation algorithm, it is also necessary to provide (in the working directory) comma separated value files that contain experimental data from which model input parameters will be estimated (unless synthetic experimental data are used).

The Gpyro property estimation program can be run in serial or in parallel, but MPI must be installed even for a serial run or the executable will not run. The above command would launch a serial run of the Gpyro property estimation program. The procedure for starting a parallel run depends on your operating system and the installed distribution and version of MPI. The following command would launch a parallel run on 16 processors under linux:

```
mpirun –np 16 –machinefile ./machines.linux /usr/local/bin/gpyro_propest
```

The syntax may be slightly different for different MPI distributions or under Windows.

The pre-compiled Windows executables are built against MS-MPI v7 and optimized for Intel processors (with the /QaxCORE-AVX2 flag in the Makefiles). If you have a different Windows MPI implementation you may have to compile the code yourself. Additionally, the Makefiles include the -axCORE-AVX2 flag when building linux binaries. Consequently, if you have a different processor architecture you may have to compile Gpyro yourself to get it to run.

2.6 Running Gpyro as a boundary condition in NIST's Fire Dynamics Simulator

Gpyro can be applied as a boundary condition in NIST's Fire Dynamics Simulator (coupling is currently to FDS Version 6.2.0). See sample 11 for an example of using Gpyro and FDS to model the combustion of a wooden crib. The Gpyro boundary condition is invoked from within FDS by setting `GPYRO = .TRUE.` on an FDS `&SURF` line that is assigned to an `OBST` or `VENT`. As an example, the following FDS `&SURF` and `&OBST` lines (from a 2D simulation) specify that Gpyro should be applied on the `OBST` at $z = 0$ m that extends from $x = 0.060$ to $x = 0.185$ m:

```
&SURF ID    = 'WHATEVER'
      COLOR = 'GREEN'
      GPYRO = .TRUE. /

&OBST XB= 0.060, 0.185, -0.01, 0.01, 0.00, 0.00, SURF_ID='WHATEVER' /
```

For coupled FDS/Gpyro simulations, it is recommended that the user start with Sample #11 and adapt it accordingly.

In order to run a coupled FDS/Gpyro simulation, the user should create two input files in the same directory. The first is a standard FDS input file (with a few modifications described later in Section 3.4), let's call it `fds_input_file.fds`. The second is a Gpyro input file called `gpyro.data` (for now, the coupled FDS/Gpyro program looks for a Gpyro input file with the hardcoded name

`gpyro.data`). A coupled FDS/Gpyro simulation is launched just as a straight FDS simulation would be:

```
fds6_mpi_gpyro.exe fds_input_file.fds
```

As mentioned above, the coupled FDS/Gpyro program assumes that the Gpyro input file is named `gpyro.data`, so it is not necessary to specify the Gpyro input file name at the command line.

2.7 Compiling Gpyro – Windows

The archive gpyro_x.yyy.zip and the Subversion repository contain a directory named "vc8" (Visual C 8). Within this directory are Visual Studio .net 2005 Solution files and a directory structure that allows one to use the Microsoft Integrated Development Environment for modifying and debugging Gpyro. If it is desired to use this development environment, then Visual Studio .net 2005 and Intel Visual Fortran 9.1 must be installed. After successfully installing Visual Studio and Intel Fortran, the Gpyro solution file can be opened by double-clicking on the gpyro.sln file from the Windows Explorer.

The working directory for debugging in Visual Studio is specified as an absolute, rather than relative, path located off of c:\gpyro\vc8\. Consequently, if the gpyro_x.yyy.zip file is unzipped to any location other than c:\gpyro, then it will be necessary to change the working directory for debugging (Within Visual Studio, go to Project → Properties → Working directory).

Since the Gpyro property estimation program uses message passing libraries for parallelization, an MPI implementation (such as MPICH1) must be correctly installed in order to compile the property estimation program. The Visual Studio .net 2005 Solution files are set up to point to an MPICH1 include directory and the MPICH1 libraries.

The linker encounters an error when compiling the coupled FDS/Gpyro executable file in a release configuration. However, it compiles successfully in the debug configuration.

2.8 Compiling Gpyro – Linux

The archive `gpyro_x.yyyy.zip` and the Subversion repository contain relative directories named `linux/gpyro`, `linux/gpyro_propest`, and `linux/gpyro_fds`. These directories contain, respectively, Makefiles for the standalone Gpyro executable, the Gpyro property estimation program, and the coupled Gpyro/FDS executable. Each Makefile is set up to use several different compilers (selectable by the user), with compiler flags based on the FDS Makefile. For example, to compile a 64 bit standalone executable for Linux using the Intel fortran compiler, navigate to `linux/gpyro`, and type:

```
make -f ../../Makefile_gpyro intel_linux_64
```

The command "make" executes the Gnu make program to build binaries, and intel_linux_64 is a label contained in the Makefile that tells make which compiler and flags to use. The user is

encouraged to peruse the Makefile to locate the correct compiler/processor combination and specify the analogous label as a command line argument to make.

Since there is a strong possibility that your compiler/processor combination may not already be set up in the Makefiles, you may have to add a new section to each Makefile to properly compile Gpyro. The Gpyro property estimation program and coupled Gpyro/FDS program can be compiled in the same way as the standalone Gpyro executable. Note that a correctly installed MPI implementation is required to compile the Gpyro property estimation program and the coupled Gpyro / FDS program.

To automate the process of building binaries, the linux/ directory contains shell script files called `make_gnu.sh` and `make_intel.sh`. To run this one of these scripts, type:

```
sh make_gnu.sh
```

Before doing so, you will probably want to modify the make.sh file to specify your compiler type and to copy the newly-compiled binaries to a folder that is in your PATH. I keep the Gpyro binaries in `/usr/local/bin`, and the shell script is set up to copy the binaries to this location, rename them to the current version, and leave a copy behind in the compile directory.

3.0 INPUT FILES

Gpyro reads plain ASCII text files that consist of a sequence of Fortran namelist groups (*e.g.*, `&GPYRO_GENERAL`) and entries associated with each Namelist group (*e.g.*, `P0 = 101325.`). The available Namelist groups and entries for each Namelist group are given in Appendix A.

As described in Section 2, a standalone Gpyro simulation requires a single input file that contains the thermophysical properties, reaction kinetics, boundary conditions, *etc*.. The property estimation program requires a single input file named `gpyro.data`, plus additional input files containing the experimental data to be read in by the property estimation algorithm for each experimental measurement that is to be used in the property estimation process. A coupled Gpyro/FDS simulation requires two input files: one FDS input file, and one input file named `gpyro.data` that is analogous to the input file for a standalone Gpyro simulation.

3.1 Spreadsheet-based "front end"

A simple user interface (or front end) is implemented though a Microsoft Excel spreadsheet. This spreadsheet uses Visual Basic macros to automatically generate ASCII input files that are read in by Gpyro. This allows the user to specify material properties, reaction kinetics, *etc*. in spreadsheet form without being concerned with the syntax of the actual input files read in by Gpyro. The spreadsheet-based front end writes a single input file that contains the Namelist groups necessary to run the Gpyro standalone executable and the property estimation executable. If the user prefers, it is possible to circumvent the spreadsheet-based front end altogether and work directly with ASCII input files.

From anywhere within the Excel-based front end, the input files required to run Gpyro can be generated via the keystroke ctrl-g (for "go"). This executes a macro that writes input files to the directory containing the Excel-based front end. Any input files that exist in the current directory may be over-written. Although input files are usually created in a Windows environment, they are compatible with Linux file systems (no issues with dos/Unix carriage return/line feed conventions have been encountered) so calculations can also be performed under *nix systems. Linux, instead of Windows, is recommended for running the material property estimation program on multiple processors. See Chapter 2 for instructions on how to run the standalone Gpyro program and the Gpyro property estimation program after the ASCII input files have been generated.

In this Users' Guide, input parameters are referenced by their name as it appears in a Namelist group. Since this name is not necessarily the same as the label in the Excel-based front end, the tables in Appendix A list both the name of a particular input parameter in a Namelist group, and the location of this parameter in the Excel-based front end file. In this way, Appendix A can be used to cross reference the discussion in the Users Guide with the spreadsheet-based front end.

3.2 Format of input files for standalone simulations

The Namelist groups specific to standalone simulations, and the analogous worksheet name in the spreadsheet-based front end, are listed in Table 1. Those Namelist groups specific to the property estimation algorithm are discussed in Section 3.3.

Table 1. Gpyro Namelist groups, associated worksheet in spreadsheet-based front end, and description of Namelist group entries.

Namelist group name	Worksheet name	Description
&GPYRO_GENERAL	general	Miscellaneous/general parameters
&GPYRO_OUTPUT	output	Point output, Profile output, and Smokeview output
&GPYRO_SPROPS	sprops	Thermophysical properties of each solid (condensed phase) species. Includes k, ρ, c, ε, κ, T_m, γ, K, ρ_{s0}.
&GPYRO_RXNS	rxns	Condensed phase (heterogeneous) reactions. Z, E, ΔH_{sol}, ΔH_{vol}, χ, n, n_{O2}
&GPYRO_GPROPS	gprops	Gas properties (c_{pg}, Lennard–Jones parameters, initial conditions, etc.)
&GPYRO_GYIELDS	gyields	Heterogeneous reactions gaseous yields matrix ($y_{s,j,k}$)
&GPYRO_HGRXNS	hgrxns	Homogeneous gaseous reactions parameters (p, q, b, Z, E, ΔH)
&GPYRO_HGYIELDS	hgyields	Homogeneous reactions gaseous yields matrix ($y_{g,j,\ell}$)
&GPYRO_IC	IC	Initial conditions
&GPYRO_GEOM	geom	Geometry
&GPYRO_ALLBC	BC	Boundary conditions
&GPYRO_CASES	cases	Number of cases to run, run time, etc.

Standard Fortran Namelist group conventions apply. The first character of a Namelist group must be an '&', and the last character must be a '/'. An input file consists of a series of Namelist groups. The order in which the Namelist groups appear in the input file does not matter, but a Namelist group may appear only once in an input file. For example, the following is valid:

```
&GPYRO_GENERAL TAMB          = 293.
               P0            = 101325. /
```

But the following is not valid:

```
&GPYRO_GENERAL TAMB          = 293. /
&GPYRO_GENERAL P0            = 101325. /
```

It is necessary to specify only parameter values that are different from Gpyro's default values. Available Namelist groups, valid entries for each Namelist group, and the default value of each entry are given in Appendix A (along with the location of each variable in the spreadsheet-based front end). One way to create a valid input file is to use the spreadsheet-based front end to generate a sample input file, and then modify it accordingly for the particular case under consideration. However, the spreadsheet-based front end uses minimal formatting, so the input files that it generates are not always easy to read.

3.3 Format of input files for material property estimation program

The input file for the material property estimation program is the same as for the standalone Gpyro input file discussed above, plus three additional Namelist groups. Whereas the input file for the standalone Gpyro program can have any file name, the input file for the material property estimation program must be named `gpyro.data` (this simplifies running in parallel with MPI). The three additional Namelist groups that must be specified in this input file (and the analogous worksheet name in the spreadsheet-based front end) are listed in Table 2.

Table 2. Gpyro property estimation Fortran Namelist groups, associated worksheet in spreadsheet-based front end, and description of Namelist group entries.

Namelist group name	Worksheet name	Description
&GA_GENINPUT	GA_GenInput	General/miscellaneous parameters
&GA_PHI	GA_Phi	Fitness weightings
&GA_VARS	GA_Vars	Variables to be optimized

An additional input file for each experimental measurement to be used in the property estimation process is required. The naming convention for these files is `exp_xx_yy.csv` where xx corresponds to IPHI (more on this in the fitness weightings section) and yy is ICASE (the index corresponding to the case number being simulated). The format of these input files is as follows:

```
NDT
t1, y1
t2, y2
t3, y3
.
.
.
```

where t_n is the n^{th} time point and y_n is the experimental quantity (mass loss rate, temperature, or thickness) at that time. Mass loss rates must be specified in units of g/m^2-s, temperatures in degrees Celsius, and thickness in meters. For example, the following is a valid experimental data file containing five temperature points:

```
5
0.0, 27.0
5.0, 50.0
30.0, 200.0
60.0, 400.0
500.0, 450.0
```

These experimental data files can be generated automatically by pasting data into the spreadsheet based front-end file. Worksheet ‘01’ corresponds to ICASE 1, worksheet ‘02’ corresponds to ICASE 2, and so on. Additional details are given in Chapter 13.

3.4 Format of input files for coupled Gpyro/FDS simulations

As mentioned above, two input files are required for a coupled Gpyro/FDS simulation: an FDS input file (any file name) and a Gpyro (standalone) input file (`gpyro.data`). The Gpyro input file is equivalent to a standalone input file.

The FDS input file is identical to an FDS 6.2.0 input file as described in the NIST documentation, with two exceptions. First, an additional keyword was added to FDS's `&SURF` Namelist group: to specify that a particular surface should be calculated using Gpyro instead of a boundary condition calculated by FDS, specify `GPYRO = .TRUE.` on the analogous `&SURF` line. The second difference is that the parameters `TIGNSTART`, `TIGNSTOP`, `IGNPOW`, and `IGNLOC` were added to the `&MISC` Namelist group. These keywords are used to add a volumetric heat source (i.e., an igniter) to an FDS calculation. The igniter has zero power output before time `TIGNSTART`, zero output after `TIGNSTOP`, and it has power output of `IGNPOW` (units of W/m^3) between `TIGNSTART` and `TIGNSTOP`. The location of the igniter is given by the sextuplet `IGNLOC`, which specifies the location a parallelepiped over which the volumetric heat source is located. See Sample #11.

THIS NEEDS TO BE UPDATED TO REFLECT CURRENT STATUS

4.0 SETTING UP 0D OR MULTIDIMENSIONAL SIMULATIONS

Gpyro can be used for 0D, 1D, 2D, and 3D simulations.

An 0D simulation corresponds to a single homogeneous particle with negligible gradients of temperature (Bi << 1) and species. This is an idealized model of thermal analysis experiments such as TGA and DSC. Quantities such as particle temperature, mass, and species mass fractions vary temporally but not spatially. In 0D simulations, the particle temperature is specified as a function of time (initial temperature plus a constant linear ramp rate).

A 1D simulation corresponds to a configuration in which lateral variations in temperature, density, and species mass fractions are small in comparison to longitudinal variations. For example, a Cone Calorimeter experiment in which a material with a thickness of 1 cm and a cross section of 10 cm by 10 cm is exposed to an approximately uniform heat flux can be reasonably approximated as 1D because gradients perpendicular to the irradiated surface are much greater than gradients parallel to the irradiated surface. This is the basic configuration that was contemplated throughout much of the Gpyro development process.

2D simulations are necessary when gradients in two directions are of comparable magnitude. For example, an ember dropped onto a porous combustible fuel bed gives rise to lateral gradients in temperature and species in the porous target fuel, and a 1D representation of this scenario is questionable.

3D simulations are necessary for complex objects or when gradients in one or more directions are not negligible

The following sections explain how to set up each type of calculation.

4.1 Batch mode vs. single run (cases worksheet and &GPYRO_CASES)

Under certain circumstances, batch mode can be used to run several related cases in succession from a single instance of Gpyro. In batch mode, `&GPYRO_CASES` (cases worksheet, see Table 3) specifies each different case to run. Batch mode can be used to change material thickness, applied heat flux level, ambient gaseous mass fractions, front-face heat transfer coefficient, ignition time, flame heat flux, thermophysical properties, reaction kinetics, etc. between multiple cases.

The number of cases to run in batch mode is specified via the `NCASES` keyword in the `&GPYRO_CASES` Namelist group (or cell A1 of the cases worksheet). A particular case is referred to by the integer ICASE.

4.2 0D calculations

A 0D simulation is specified by setting a flags in the `&GPYRO_CASES` Namelist group (or in the cases worksheet). For each ICASE, set the value of `ZEROD` to `.TRUE.` (Column D in the spreadsheet-based front end). The linear heating rate, in units of °C/min is specified by the

keyword BETA (Column E). For example, the following &GPYRO_CASES line specifies two TGA/DSC experiments with linear ramp rates of 10 °C/min and 20 °C/min that run out to 3600 s and 1800 s, respectively:

```
&GPYRO_CASES NCASES   = 2
             ZEROD(1) = .TRUE.
             TSTOP(1) = 3600.
             BETA (1) = 10.
             ZEROD(2) = .TRUE.
             TSTOP(2) = 1800.
             BETA (2) = 20.   /
```

Note from Table 3 that additional keywords are part of the &GPYRO_CASES line; however, they play no role in a 0D calculation, so it is not necessary to specify them for a 0D calculation.

4.3 1D calculations

NEED TO UPDATE THIS SECTION

4.4 2D and 3D calculations

NEED TO UPDATE THIS SECTION

5.0 GEOMETRY AND INITIAL CONDITIONS

NEED TO UPDATE THIS TO CURRENT CODE

6.0 BOUNDARY CONDITIONS

Boundary conditions are not necessary for 0D calculations, since Gpyro solves coupled transient ordinary differential equations that only require specification of initial conditions.

THIS SECTION NEEDS TO BE UPDATED TO GENERALIZED BOUNDARY CONDITION SPECIFICATION

7.0 CONDENSED-PHASE THERMOPHYSICAL PROPERTIES

Condensed-phase thermophysical properties are specified through the &GPRYO_SPROPS Namelist group (sprops stands for solid properties). The available keywords, and their default values, are listed in Table 7. The reader not familiar with Gpyro should refer to Chapter 2 of the Gpyro Technical Reference [1].

The keyword NSSPEC is the number of solid species for which properties will be specified. Each solid species should be assigned a unique name (character string) via the keyword NAME, *e.g.* wood, char, ash, glass_fibers, etc. Do not use spaces in the names (underscores can be used instead).

As discussed in Chapter 2 of the Gpyro Technical Reference [1], the presumed temperature dependency for thermal conductivity, specific heat capacity, and density is $\phi(T) = \phi_0 (T/T_r)^{n_\phi}$ where ϕ represents k, c, or ρ and T_r is the reference temperature (specified on the &GPYRO_GENERAL line). The relevant keywords are K0Z and NKZ for thermal conductivity in the z direction, C0 and NC for specific heat capacity, and R0 and NR for density. Note that K0X and NKX are the keywords controlling thermal conductivity in the x direction, but the coding for anisotropic materials has not yet been completed so for the time being, K0X and NKX have no effect on the calculation because the thermal conductivity in the x direction is assumed to be the same as in the z direction. The units of thermal conductivity are W/m-K, the units of specific heat capacity are J/kg-K, and the units of density are kg/m^3.

The meaning of density and thermal conductivity specified in &GPYRO_SPROPS is different depending on the value of SOLVE_POROSITY on the &GPYRO_GENERAL line. By default, SOLVE_POROSITY = .FALSE., and the user-specified thermal conductivity of a particular species is an effective value; similarly, the user-specified density of a species is its bulk density. In this formulation, porosity is treated as a property of a condensed-phase species, and the effective thermal conductivity and bulk density both already include the effect of porosity. This is the formulation that was used throughout Ref. [2]. However, if the user specifies SOLVE_POROSITY = .TRUE., then porosity is no longer a property of a species. Instead, it is calculated from the condensed-phase mass conservation equation assuming no volume change (shrinkage or swelling) occurs. Additionally, with this formulation, the user-specified thermal conductivity and density are interpreted as those of a nonporous solid. With this formulation, the thermal conductivity that appears in the condensed-phase energy conservation equation is $\bar{k} = (1-\psi)\overline{k_s}$ where ψ is porosity and $\overline{k_s}$ is the weighted thermal conductivity of the solid assuming it is nonporous. Similarly, with this formulation, the bulk density is calculated as $\bar{\rho} = (1-\psi)\overline{\rho_s}$ where $\overline{\rho_s}$ is the weighted density of the solid assuming it is nonporous. Details are given in the Technical Reference Guide [1].

The emissivity of each species is specified by the keyword EMIS. The effective surface emissivity is calculated as a volume-weighted average using the species volume fractions in the boundary cell. Thus, Gpyro can account for the change in the emissive/absorptive characteristics that some materials exhibit as they pyrolyze (for example, the darkening of wood). Another important

radiative property is KAPPA, the absorption coefficient κ (m^{-1}). As with emissivity, the effective absorption coefficient used in the calculations is also a volume–weighted quantity. If the value of κ in an irradiated boundary cell is greater than 10^6 m^{-1}, then Gpyro considers only surface absorption. Otherwise, Gpyro calculates in-depth radiation absorption ($\partial \dot{q}_r'' / \partial z$) as described in Section 4.1.6 of the Gpyro Technical Reference [1]. In addition to calculating in-depth radiation absorption, Gpyro can also simulate radiative heat transfer across pores. The parameter GAMMA (γ, units of m) controls the radiative thermal conductivity, see Equation 7 of the Gpyro Technical Reference [1]. Essentially, for nonzero values of γ, a species is given a component of its effective thermal conductivity that varies proportional to T^3.

The parameters that control melting are TMELT (T_m), DHMELT (ΔH_m) and SIGMA2MELT (σ_m^2), see Equation 9 of the Gpyro Technical Reference [1]. TMELT is the melting temperature (K), DHMELT is the latent heat of melting (J/kg), and σ_m^2 (units of K^2) is the parameter in Equation 9c of the Gpyro Technical Reference [1] that controls the width of the Gaussian peak over which the latent heat of melting is distributed as an apparent specific heat capacity. Melting is handled in a simple way through an increase in the specific heat capacity in the vicinity of the melt temperature. Other thermophysical properties (thermal conductivity, density) do not exhibit a discontinuity at the melting temperature, and a species above its melting temperature does not become fluid and flow in Gpyro.

Permeability (units of m^2) in the z direction is specified via the keyword PERMZ. Similarly, permeability in the x direction is specified via the keyword PERMX, but for the time being the x-direction permeability is the same as in the z direction because the coding to facilitate anisotropic behavior has not yet been completed. Permeability affects only the gas phase momentum conservation equation through Darcy's law.

The keyword RHOS0 is used to specify the value of ρ_{s0} (kg/m^3). As with thermal conductivity and density, the meaning of ρ_{s0} depends on the value of SOLVE_POROSITY. When SOLVE_POROSITY = .TRUE., ρ_{s0} is used to calculate the initial porosity (*i.e.*, at t = 0 s) as $\psi_0 = 1 - \overline{\rho}_0 / \overline{\rho_{s0}}$ (where a subscript 0 denotes initial). When SOLVE_POROSITY = .FALSE., ρ_{s0} is used to calculate the porosity of each individual species as $\psi_i = 1 - \rho_i / \rho_{s0,i}$ and weighted porosity is calculated as $\overline{\psi} = \sum X_i \psi_i$.

Finally, mean pore diameter (d_p, units of m) is specified by the keyword PORE_DIAMETER. Presently, the mean pore diameter affects the calculations only when a "two temperature" model is used wherein thermal equilibrium is not assumed between the condensed-phase and the gas-phase, and the volumetric heat transfer coefficient is calculated based on a Nusselt number correlation that includes the pore diameter (see further discussion in &GPYRO_GENERAL section).

A sample &GPYRO_SPROPS line is shown below:

```
&GPYRO_SPROPS NSSPEC           = 2
              NAME        (1) = 'wood'
```

```
K0Z          (1) = 0.2
R0           (1) = 400.
C0           (1) = 1500.
EMIS         (1) = 0.7
GAMMA        (1) = 0.0
PERMZ        (1) = 1D-10
RS0          (1) = 500.
NAME         (2) = 'char'
K0Z          (2) = 0.1
R0           (2) = 100.
C0           (2) = 1200.
EMIS         (2) = 0.95
GAMMA        (2) = 0.005
PERMZ        (2) = 1D-10
RS0          (2) = 500. /
```

8.0 CONDENSED-PHASE REACTIONS

Gpyro's treatment of condensed-phase reactions is described in Chapter 3 of the Gpyro Technical Reference [1]. Parameters controlling condensed-phase reactions are specified in the &GPYRO_RXNS Namelist group. The number of reactions to be considered is NRXNS. The keyword CFROM gives the name of the condensed phase reactant (the species that will be consumed) and the keyword CTO gives the name of the condensed phase product (the species that will be produced). These parameters begin with 'C' as a reminder that a character string is required; internally, Gpryo tracks reactant and product species according to their integer indices (IFROM and ITO). Using the nomenclature from the Gpyro Technical Reference [1], for reaction k the CFROM species is A_k, and the CTO species is B_k. The CFROM and CTO names must correspond exactly to the name of a condensed phase species specified in the &GPYRO_SPROPS (all names are case-sensitive). If a CFROM species specified in &GPYRO_RXNS does not have an analogous definition in &GPYRO_SPROPS, Gpyro will give an error message and shut down. However, if a CTO species is specified without an analogous definition in &GPYRO_SPROPS, Gpyro does not treat this as an error. Instead, Gpyro assumes that the reaction produces only gases, i.e. the reaction leaves behind no condensed phase residue. Be sure you do not inadvertently spell a CTO species incorrectly and end up with a noncharring reaction when you intended to specify a charring reaction!

Reaction kinetics are controlled primarily by the keywords Z (pre-exponential factor, s^{-1}), E (activation energy, kJ/mol), and ORDER (reaction order, dimensionless). Reaction rates are zero for temperatures below TCRIT. See Section 3.4 of the Gpyro Technical Reference [1] for details. Additional kinetic models have been added to Gpyro. See Section 3.4.1 in the Gpyro Technical Reference for a discussion of the different available kinetic models. A particular kinetic model is selected by specifying the integer IKINETICMODEL. For those kinetic models that require two exponents, the user should also specify the value of m via the keyword M. For kinetic model type 9 (catalytic reaction) the user must also specify KCAT (K_{cat} in Section 3.4.1 of the Gpyro Technical Reference) and ICAT (the index of the condensed-phase catalytic species).

Heats of reaction are specified via the keywords DHS and DHV. The latent enthalpy difference between the condensed phase product species and the condensed phase reactant species (units of J/kg) is DHS, analogous to ΔH_{sol} in Equation 50 of the Gpyro Technical Reference [1]. A positive value gives an endothermic reaction and a negative value gives an exothermic reaction. DHV (units of J/kg) is analogous to ΔH_{vol} in Equation 50 of the Gpyro Technical Reference [1]. This parameter is sometimes called the heat of volatilization, heat of vaporization, or heat of pyrolysis. It is the amount of heat required to volatilize unit mass of the condensed phase reactant at whatever temperature this volatilization occurs. A positive value of ΔH_{vol} implies an endothermic reaction, and a negative value implies an exothermic reaction. DHS and DHV are specified on a Joule per kilogram of reactant consumed basis. That is, ΔH_{vol} has units of Joules per kilogram of gases liberated from the condensed phase, and ΔH_{sol} has units of Joules per kilogram of condensed phase species B formed from condensed phase species A. A heat of reaction with units of Joules per kilogram of species A destroyed can be recovered by setting $\Delta H_{sol} = \Delta H_{vol}$. See the discussion after Equation 50 of the Gpyro Technical Reference [1]. A slightly different treatment of heats of

reaction can be obtained by setting CONSTANT_DHVOL = .FALSE. on the &GPYRO_GENERAL line. In this case, the parameter DHV is interpreted as $\Delta H^{\circ}_{vol,k}$ and, as described in Section 3.5 of the Gpyro Technical Reference [1], the heat of reaction is calculated as $\dot{Q}'''_{s,k} = -\dot{\omega}'''_{fB_k} \Delta H_{sol,k} - \dot{\omega}'''_{fg_k} \Delta H^{\circ}_{vol,k} - \dot{\omega}'''_{fg_k} \left(h_{g_k} - h_{A_k} \right)$.

Intumescent reactions can be specified by setting the value of χ (keyword CHI) to a value between 0 and 1. As discussed in Sections 3.1 and 3.3 of the Gpyro Technical Reference [1], χ is the fraction of the density difference between the condensed phase reactant species and the condensed phase product species that is realized as gases. A value of $\chi = 1$ means that all of the density difference caused by the formation of a lower density solid from a higher density solid will result in generation of gaseous species, and a value of $\chi = 0$ means that no gases will be generated and swelling will occur to conserve mass. Values of χ between 0 and 1 cause intumescence (swelling) to occur simultaneously with the release of gases. For cases where SOLVE_POROSITY = .TRUE., the value of χ must be 1.0.

The reaction order in the local oxygen mass fraction (n_{O2}, see Equations 46 and 47 in the Gpyro Technical Reference [1]) is specified via the ORDERO2 keyword. The user should set $n_{O2} = 0$ for reactions that are not affected by the local oxygen concentration inside the decomposing solid. The parameter IO2TYPE affects how oxidative reactions are treated. When IO2TYPE = 0, the reaction rate is proportional to $\left(1 + Y_{O_2}\right)^{n_{O2}}$; otherwise the reaction rate is proportional to $Y_{O_2}^{n_{O2}}$ (see Equation 47 in the Gpyro Technical Reference [1]).

A sample &GPYRO_RXNS line is given below, and it is assumed that the &GPYRO_SPROPS line is as given above. The first reaction is the endothermic thermal pyrolysis of wood to form char; and the second reaction is the exothermic oxidation of char to form gases (and no solid residue), which would cause surface regression.

```
&GPYRO_RXNS NRXNS       = 2
            CFROM   (1) = 'wood'
            CTO     (1) = 'char'
            Z       (1) = 5D9
            E       (1) = 130.
            DHV     (1) = 1D6
            CHI     (1) = 1.0
            ORDER   (1) = 1.0
            CFROM   (2) = 'char'
            CTO     (2) = 'gases'
            Z       (2) = 1D12
            E       (2) = 200.
            DHV     (2) = -1D6
            CHI     (2) = 1.0
            ORDER   (2) = 1.0
            ORDERO2 (2) = 1.0
            IO2TYPE (2) = 0   /
```

In addition to specifying the parameters that control reaction kinetics and thermodynamics, the user must also specify the "yields" of gaseous species produced or consumed by that reaction. As explained in Section 3.1 of the Gpyro Technical Reference [1], $y_{s,j,k}$ is the "species yield matrix". It is specified in Gpyro via the `&GPYRO_GYIELDS` Namelist group, which consists of a single keyword `GYIELDS(IGSPEC,IRXN)` where IGSPEC is the integer index of the gaseous species that is consumed or produced and IRXN is the integer index of the condensed-phase reaction. That is, each row corresponds to a gaseous species (specified in `&GPYRO_GPROPS`), and each column corresponds to a condensed phase reaction (specified in `&GPYRO_RXNS`). As an example, the entry in the third row, second column is the yield for the third gaseous species (as specified in `&GPYRO_GPROPS`) from the second condensed phase reaction (as specified in `&GPYRO_RXNS`). To conserve mass, all entries in a column must add to 1. Note that entries may be either positive, negative, or zero, with a positive entry corresponding to production of a gaseous species, and a negative entry corresponding to consumption of a gaseous species. Gpyro tracks the composition of the volatiles inside a decomposing solid (and escaping from its surface).

The following is a sample `&GPYRO_GYIELDS` line where it is assumed that the `&GPYRO_RXNS` line specified above applies, and that gaseous species 1 is 'pyrolysate', gaseous species 2 is 'oxygen', gaseous species 3 is 'nitrogen', and gaseous species 4 is 'products':

```
&GPYRO_GYIELDS GYIELDS(1,1) =  1.0
               GYIELDS(1,2) =  0.0
               GYIELDS(2,2) = -1.0
               GYIELDS(3,2) =  0.0
               GYIELDS(4,2) =  2.0 /
```

Note that reaction 1 produces only gaseous species 1 (pyrolysate), whereas reaction 2 consumes 1 kg of gaseous species 2 (oxygen) and produces 2 kg of gaseous species 4 (products). Reaction 2 does not consume or produce gaseous species 1 (pyrolysate) or gaseous species 3 (nitrogen). Each of these four species must have a corresponding entry on the `&GPYRO_GPROPS` line. Note that reaction 2 will not occur unless oxygen is present.

9.0 GAS-PHASE THERMOPHYSICAL PROPERTIES

Parameters controlling gas-phase thermophysical properties are specified via the &GPYRO_GPROPS Namelist group. The number of gaseous species is specified by the keyword NGSPEC, which should be set to a minimum of 1, even if the gaseous species conservation equation is not being explicitly solved. IBG is the gaseous species index of the background species, and IO2 is the gaseous species index corresponding to oxygen. IBG and IO2 are needed because the diffusion coefficient is calculated from Chapman-Enskog theory as the diffusivity of oxygen into the background species (all species are assigned the same diffusivity). IO2 is also used by Gpyro for heterogeneous reactions that are affected by the local oxygen concentration, i.e. reactions for which $n_{O2} \neq 0$, see Equations 46 and 47 in the Gpyro Technical Reference [1]. The gas-phase specific heat capacity (in units of J/kg-K) is specified via the keyword CPG. For simplicity, it is assumed that all gases have the same specific heat capacity, independent of temperature. Each of these parameters should have only a single entry on the &GPYRO_GPROPS line.

Several keywords must be specified for each gaseous species. NAME is a character string that assigns a unique name to each species (do not use spaces). YJ0 is the initial mass fraction of each species (which is assumed to be spatially invariant). The keyword M is the molecular weight of each species in units of g/mol. SIGMA and EPSOK are the Lennard Jones model parameters σ and ε/k that determine species' diffusivity, see Equation 19 in the Gpyro Technical Reference [1]. Note that σ is specified in units of Å, and ε/k has units of K. As described in the Gpyro Technical Reference [1], all species are assumed to have the same diffusivity, taken as that of oxygen into the background species. Thus, Gpyro only uses the values σ and ε/k for the background species (specified by IBG) and oxygen (specified by IO2). The values of these parameters for the other species do not affect the calculations, but must be specified to avoid i/o errors.

The parameters B, TONSET, and TSPAN are remnants from a simplified method of treating char oxidation. Although the code sections still remain, they have not been tested or verified and the user should set B = 0 (or simply not specify a value for B since the default value is 0); if this is done then the values of TONSET and TSPAN won't affect the calculations.

The following is a sample &GPYRO_GPROPS line, consistent with the sample &GPYRO_GYIELDS line given in the previous section.

```
&GPYRO_GPROPS NGSPEC    = 1
              IBG       = 1
              IO2       = 2
              CPG       = 1000.
              NAME  (1) = 'pyrolysate'
              YJ0   (1) = 0.0
              M     (1) = 100.
              NAME  (2) = 'oxygen'
              YJ0   (2) = 0.23
```

```
M     (2) = 32.
NAME  (3) = 'nitrogen'
YJ0   (3) = 0.77
M     (3) = 28.
NAME  (4) = 'products'
YJ0   (4) = 0.0
M     (4) = 44. /
```

10.0 HOMOGENEOUS GAS-PHASE REACTIONS

The parameters that control homogeneous gas phase reactions are specified in the &GPYRO_HGRXNS Namelist group. The number of homogeneous gas phase reactions is specified by the keyword NHGRXNS. Homogeneous gaseous reaction rates are calculated according to Equation 54 in the Gpyro Technical Reference [1]. The character string corresponding to the name of the first reactant should be given as CREACTANT1; similarly, the character string corresponding to the name of the second reactant should be given as CREACTANT2. The reaction rate is proportional to the molar concentration of reactant 1 raised to the power P multiplied by the molar concentration of reactant 2 raised to the power Q. The reaction rate is proportional to the absolute temperature raised to the temperature exponent (B) and the Arrhenius term controlled by the pre–exponential factor (Z) and the activation energy E. Units of kg, m^3, mole, and s are used for the pre-exponential factor (Z), but for dimensional consistency the actual units of Z vary with the specified values of P, Q, and B. For P = 1, Q = 1, and B = 0, the units of Z are kg-m^3/$mole^2$-s. The activation energy E has units of kJ/mol. The heat of reaction per unit mass of reactant 1 consumed (units of J/kg) is specified via the keyword DH. This corresponds to ΔH_ℓ in Equation 60 of the Gpyro Technical Reference [1]. Note that ΔH_ℓ is negative for exothermic reactions and positive for endothermic reactions.

The following is a sample &GPYRO_HGRXNS line:

```
&GPYRO_HGRXNS NHGRXNS       =  1
              CREACTANT1(1) =  'pyrolysate'
              CREACTANT2(1) =  'oxygen'
              P         (1) =  1.0
              Q         (1) =  1.0
              B         (1) =  0.0
              Z         (1) =  1D10
              E         (1) =  195.0
              DH        (1) = -15D6 /
```

Homogeneous gaseous reaction rates are translated to formation or consumption rates of individual gaseous species by the “homogeneous gaseous species yield matrix” (see Equation 53 in the Gpyro Technical Reference [1]). The homogeneous gaseous reactions species yield matrix is specified in a similar way via the &GPRYO_HGYIELDS Namelist group. The only entry available on the &GPRYO_HGYIELDS line is HGYIELDS(IGSPEC, IRXN) where IRXN is the index of the homogeneous gaseous reaction and IGSPEC is the index of the gaseous species formed or consumed by that reaction. The specification of the yields for a homogeneous gaseous reaction is similar to the method for heterogeneous reactions, except here the sum of any column must add to zero (instead of 1). That is, there can be no net creation of gaseous mass by a homogeneous gas phase reaction. Negative entries correspond to the consumption of a species, and positive entries correspond to its production.

The following is a sample &GPYRO_HGYIELDS line which specifies that reaction 1 consumes 1 kg of gaseous species 1 and 3 kg of gaseous species 2 to produce 4 kg of gaseous species 4. This

is consistent with the preceding sample &GPROPS line wherein gaseous species 1 is ‘pyrolysate’, gaseous species 2 is ‘oxygen’, and gaseous species 4 is ‘products.

```
&GPYRO_HGYIELDS HGYIELDS(1,1) = -1.0
                HGYIELDS(2,1) = -3.0
                HGYIELDS(3,1) =  0.0
                HGYIELDS(4,1) =  4.0 /
```

11.0 OUTPUT FILES

The dump routines allow the user to tailor the amount of output data that will be dumped by Gpyro to meet his or her needs. Parameters controlling output data are specified on the `&GPYRO_OUTPUT` line. All output files are tagged with the character string given by the keyword `CASENAME`.

The three primary types of output that can be dumped by Gpyro are point quantities, profiles, and Smokeview files for visualization of 2D simulations in Smokeview. These are discussed in Sections 11.1, 11.2, and 11.3. The quantities available to dump are shown in Table 1 on the following page.

An option is included to reduce the time interval between file dumps when the temperature increases above a certain value. When the temperature (solid or gas) anywhere in the computational domain increases above `TMP_REDUCED_DTDUMP` (units of K), the time interval between dumps for all output files is reduced to `REDUCED_DTDUMP` (s). This is useful for transition to flame simulations where a long-duration, low-temperature smolder period with a large characteristic timescale is followed by a very rapid increase in temperature at the transition to flame characterized by a very short time scale. By dropping the time interval between dumps to a smaller value, it is possible to "see" the transition to flame in output files. If the timestep at any point is reduced below `DTMIN_KILL` (s), Gpyro will shut down.

Table 1. Quantities available for dump.

Name	Index meaning	Description	Units
'TEMPERATURE'		Condensed-phase temperature	ºC
'ENTHALPY'		Condensed-phase enthalpy	J/kg
'YI'	ISSPEC	Mass fraction of condensed-phase species	kg/kg
'XI'	ISSPEC	Volume fraction of condensed-phase species	m^3/m^3
'CI'	ISSPEC	Concentration of condensed-phase species	kg/m^3
'YISUM'		Sum of all condensed-phase mass fractions	kg/kg
'REACTION_RATE_K'	IRXN	Rate of k^{th} condensed-phase reaction	kg/m^3-s
'YJ'	IGSPEC	Mass fraction of gaseous species	kg/kg
'CJ'	IGSPEC	Concentration of gaseous species	kg/m^3
'YJSUM'		Sum of all gas-phase mass fractions	kg/kg
'REACTION_RATE_L'	IHGRXN	Rate of ℓ^{th} homogeneous gaseous reaction	kg/m^3-s
'S'		Source term in condensed-phase energy eq.	J/m^3-s
'THERMAL_CONDUCTIVITY'		Condensed-phase thermal conductivity	W/m-K
'BULK_DENSITY'		Condensed-phase bulk density	kg/m^3
'SPECIFIC_HEAT_CAPACITY'		Condensed-phase specific heat capacity	J/kg-K
'Z_GRID_SIZE'		Grid size in z direction	m
'PRESSURE'		Overpressure	Pa
'MASS_FLUX_TOTAL_Z'		z-direction mass flux	g/m^2-s
'MASS_FLUX_TOTAL_X'		x-direction mass flux	g/m^2-s
'GAS_TEMPERATURE'		Gas-phase temperature	ºC
'GAS_ENTHALPY'		Gas-phase enthalpy	J/kg-K
'TG-T'		Gas temperature minus solid temperature	ºC
'SHP'		Positive part of solid energy eq. source term	J/m^3-s
'SHM'		Negative part of solid energy eq. source term	J/m^3-s
'PERMEABILITY'		Permeability	m^2
'POROSITY'		Porosity	m^3/m^3
'D12'		Gaseous diffusion coefficient	m^2/s
'SHGP'		Positive part of gas energy eq. source term	J/m^3-s
'SHGM'		Negative part of gas energy eq. source term	J/m^3-s
'QSG'		Rate of heat transfer from solid to gas phase	J/m^3-s
'RYIDZSIGMA'	ISPEC	$\left(\overline{\rho} Y_i \Delta z\right)_\Sigma$	kg/m^2
'UNREACTEDNESS'	ISPEC	1 - α	-
'GOMEGA3'		Conversion rate of solid to gas	kg/m^3-s
'RE'		Reynolds number	-
'NU'		Nusselt number	-
'HCV'		Volumetric heat transfer coefficient	W/m^3-K
'M/M0'		Mass divided by initial mass in a cell	kg/kg
'MLR'[3]	IGSPEC[1]	Flux of gaseous species	g/m^2-s
'FRONT_FACE_DIFFUSIVE_FLUX'[3]	IGSPEC	Diffusive flux of gas species at front face	g/m^2-s
'FRONT_FACE_CONVECTIVE_FLUX'[3]	IGSPEC	Convective flux of gas species at front face	g/m^2-s
'FRONT_FACE_MASS_FLUX'[3]	IGSPEC	Mass flux of gas species at front face	g/m^2-s
'MPPI'[2,3]	ISPEC	Mass per unit area of condensed species	kg/m^2
'THICKNESS'[2,3]		Total thickness	m
'DT'[2,3]		Timestep	s
'N_ITERATIONS'[2,3]		Number of iterations for convergence	-

ISSPEC: Index of solid species
IGSPEC: Index of gaseous species
IRXN: Index of heterogeneous/condensed-phase reaction
IHGRXN: Index of heterogeneous gaseous reaction
[1] Specify IGSPEC = 0 for total MLR.
[2] Does not vary spatially
[3] Available only as a point quantity.

11.1 Point dumps

The user can specify the point values that Gpryo should write to disk. Point values are similar to a "thermocouple" that can be precisely positioned and configured to measure any quantity (not just temperature). Point values are written in comma separated value format to a file named `CASENAME_summary_IQE.csv`.

The number of point quantities to be dumped should be specified via the keyword `N_POINT_QUANTITIES`. For each quantity, the user then specifies `POINT_Z(I)`, `POINT_X(I)`, `POINT_QUANTITY(I)`, and sometimes `POINT_QUANTITY_INDEX(I)`. Here, `I` is an integer that ranges from 1 to `N_POINT_QUANTITIES` and `POINT_QUANTITY(I)` is a character string that specifies the quantity to be dumped. Table 1 lists the available point quantities. `POINT_Z(I)` and `POINT_X(I)` are the z and x coordinates (units of m) at which the analogous quantity will be dumped. The user is cautioned to request point dumps at positions that are slightly offset from the center of a grid cell because the subroutine that decides which cell corresponds to a requested location is crude (and will be refined in a later version). This will prevent output data from being inadvertently shifted by one cell compared to what the user expects. For example, if grid centers are at 1 mm, 2 mm, 3 mm, etc., then the user should request point dumps at 1.1 mm, 2.1 mm, 3.1 mm, and so on.

The integer `POINT_QUANTITY_INDEX(I)` applies to point quantities such as condensed-phase mass fractions (`POINT_QUANTITY(I) = 'YI'`) for which an additional index is required to specify the desired value (*e.g.*, set `POINT_QUANTITY_INDEX` equal to 3 to request the mass fraction of species 3).

This is clarified in the following sample (partial) `&GPYRO_OUTPUT` line:

```
&GPYRO_OUTPUT N_POINT_QUANTITIES       = 3
              DTDUMP_POINT             = 0.1
              POINT_QUANTITY       (1) = 'TEMPERATURE'
              POINT_Z              (1) = 0.01
              POINT_X              (1) = 0.00
              POINT_QUANTITY       (2) = 'YI'
              POINT_Z              (2) = 0.01
              POINT_X              (2) = 0.00
              POINT_QUANTITY_INDEX (2) = 1
              POINT_QUANTITY       (3) = 'YI'
              POINT_Z              (3) = 0.01
              POINT_X              (3) = 0.00
              POINT_QUANTITY_INDEX (3) = 2
              .
              .
              .                                  /
```

This line tells Gpyro to dump, at the location z = 0.01 m and x = 0.00 m, condensed-phase temperature, mass fraction of condensed-phase species 1, and mass fraction of condensed-phase

species 2. In addition to specifying the quantities to be dumped (and their location and index, if appropriate) the user should also specify the dump frequency in seconds via the keyword DTDUMP_POINT. For example, DTDUMP_POINT = 5.0 indicates that all point quantities will be dumped every 5 s.

11.2 Profile dumps

In addition to dumping single point quantities at a particular spatial coordinate, the user can also tell Gpyro to dump profiles at a given *x* or *z* location. Each profile is written to a file named CASENAME_profile_xx_QUANTITY_NAME.csv where CASENAME is the user-specified case name, xx is the profile number, and QUANTITY_NAME is the character string corresponding to the quantity to be dumped.

The relevant keywords are PROFILE_QUANTITY(I), PROFILE_QUANTITY_INDEX(I), PROFILE_TYPE(I), PROFILE_LOCATION(I), and PROFILE_ISKIP(I). As with the point quantities, I is an integer that ranges from 1 to N_PROFILE_QUANTITIES and is a character string corresponding to the quantity to be dumped. The available quantities are listed in Table 1 (as indicated by note 3, not all quantities are available for profile dumps). For 1D calculations, all available quantities can be dumped by specifying DUMP_EVERYTHING = .TRUE.. Profiles will be dumped every DTDUMP_PROFILE seconds.

PROFILE_QUANTITY_INDEX(I) is analogous to POINT_QUANTITY_INDEX(I) discussed in the previous section. Specifying the location of a profile is a bit different than specifying the location of a point quantity. The keyword PROFILE_TYPE tells Gpyro whether a *z* or *x* profile is desired. For profiles in the *z* direction, specify PROFILE_TYPE = 'z'; for profiles in the *x* direction, specify PROFILE_TYPE = 'x'. The user must also specify via PROFILE_LOCATION the *x* or *z* coordinate at which the profile will be recorded. For PROFILE_TYPE = 'z', the value of PROFILE_LOCATION specifies the *x* coordinate at which the profile is recorded. Similarly, for PROFILE_TYPE = 'x', the value of PROFILE_LOCATION specifies the *z* coordinate at which the profile is recorded.

The parameter PROFILE_ISKIP(I) controls the fraction of the cells that get dumped to disk. For example, if PROFILE_ISKIP(I) is set to 1, then the value in every cell gets written to disk; if it is set to 5, then every fifth cell gets dumped, and so on. This is useful if, for example, you are using a 0.1 mm grid resolution but you are really only interested in seeing output data to the nearest 1 mm, in which case you would specify PROFILE_ISKIP(I). This parameter can also be used to ensure that all output fits within the 256 column limitation of older versions of MS Excel.

A sample (partial) &GPYRO_OUTPUT line is given below:

```
&GPYRO_OUTPUT N_PROFILE_QUANTITIES = 2
              DTDUMP_PROFILE       = 1.0
              PROFILE_QUANTITY (1) = 'TEMPERATURE'
              PROFILE_TYPE     (1) = 'z'
              PROFILE_LOCATION (1) = 0.02
```

```
PROFILE_ISKIP    (1) = 1
PROFILE_QUANTITY (2) = 'TEMPERATURE'
PROFILE_TYPE     (2) = 'x'
PROFILE_LOCATION (2) = 0.05
PROFILE_ISKIP    (2) = 10
.
.
.
```

This line tells Gpyro to dump two temperature profiles at an interval of 1.0 s. The first profile will be dumped in the z direction at $x = 0.02$ m and the value in every cell will be dumped. The second will be dumped in the x direction at $z = 0.05$ m and the value in every tenth cell will be dumped.

11.3 Smokeview visualization files (2D calculations only)

For 2D calculations only, Gpyro can dump output files in NIST Smokeview format, making it possible to visualize the calculations with Smokeview. This allows 2D transient slices of temperature, density, thermal conductivity, etc. to be visualized with Smokeview by double-clicking on the .smv file that is created by Gpyro. The .smv file is named `CASENAME.smv`, and Smokeview slice files are names `CASENAME_xx.sf` where xx corresponds to the slice file number.

The user should specify the number of Smokeview quantities to dump via the keyword `N_SMOKEVIEW_QUANTITIES` as well as the time interval (s) between Smokeview dumps with the keyword `DTDUMP_SMOKEVIEW`. The quantity to be dumped is specified with the keyword `SMOKEVIEW_QUANTITY(I)`. Available quantities are listed in Table 1 (as indicated by note 3, not all quantities are available for Smokeview dumps). The keyword `SMOKEVIEW_QUANTITY_INDEX(I)` is analogous to the quantities `POINT_QUANTITY_INDEX(I)` and `PROFILE_QUANTITY_INDEX(I)` described above.

The following is a sample (partial) `&GPYRO_OUTPUT` line that requests that Gpyro write a Smokeview file for condensed-phase temperature, with a dump interval of 1 s.

```
&GPYRO_OUTPUT N_SMOKEVIEW_QUANTITIES      = 1
              DTDUMP_SMOKEVIEW            = 1.0
              SMOKEVIEW_QUANTITY(1)       = 'TEMPERATURE'
              .
              .
              .                                          /
```

11.4 Changing the dump interval mid-simulation

In some situations, it may be desirable to change the interval at which Gpyro writes to disk. For example, the time scale associated with smolder propagation is much longer than the timescale associated with transition from smolder to flame. Consequently, disk dumps at an interval of 1 s may be appropriate during smolder propagation, but a smaller dump interval (e.g., 0.001 s) would

be needed to “see” the transition to flame. For this reason, an option was added wherein the dump interval can be reduced when the temperature (solid or gas) increases above a certain value. More specifically, when the gas or solid temperature increases above `TMP_REDUCED_DTDUMP` (units of K), all output files are written to disk at an interval of `REDUCED_DTDUMP` (units of s). Additionally, if the timestep is reduced below `DTMIN_KILL` (units of s) the simulation shuts down gracefully. The same is done if NaN, -Infinity, or +Infinity occur.

11.5 .out file

Gpyro automatically generates the file `CASENAME.out`, which contains a print out of all user-specified input parameters. If something in a simulation seems anomalous, it is a good idea to check this file to ensure that there are no input errors (for example, misspelling a Namelist group) and that Gpyro is using the input parameters that the user intended. During run time, Gpyro also prints a few summary quantities to `CASENAME.out`; these are useful for monitoring simulation progress.

11.7 CPU timing file

If a simulation runs to completion, Gpyro dumps `CASENAME_timing.csv`. This file lists the CPU time used by the main subroutines. This is probably more important for debugging and code optimization than general code usage.

12.0 MISCELLANEOUS PARAMETERS

Miscellaneous parameters are specified via the &GPYRO_GENERAL Namelist group. The available keywords are discussed below, and default values are given in Table 8.

An important parameter to set is DT0, the initial time step (s). If Gpyro cannot obtain a converged solution within the user-specified number of iterations, it will automatically reduce the timestep until a converged solution is obtained.

12.1 Environmental variables – TAMB, TREF, P0, GX, GZ

The keyword TAMB is used to specify the ambient temperature in units of K. It is relevant only when USE_BC_PATCHES = .FALSE.. The reference temperature (T_r, K) that appears in the presumed temperature dependency of condensed-phase properties is TREF. The background pressure is specified via P0 (units of Pa). For cases where pressure is not explicitly solved (SOLVE_PRESSURE = .FALSE.), the pressure is assumed uniform and equal to P0. For 1D calculations where the pressure is solved but USE_BC_PATCHES = .FALSE., the pressure at $z = 0$ is equal to P0.

The parameters GX and GZ are the x and z components of the gravity vector in units of m/s^2. They influence 2D simulations by inducing a buoyant flow. For now, only GZ has any effect on the calculations, *i.e.* it is assumed that the gravity vector has a component only in the z direction. GZ only has an effect when USE_BC_PATCHES = .TRUE.. Buoyancy is included in 2D simulations when GZ is set to a value other than 0, and the density at infinity is calculated at temperature TAMB using the gas compositions specified in the &GPYRO_QE Namelist group. A positive value of GZ corresponds to a gravity vector pointing in the $+z$ direction.

12.2 Gas/solid heat transfer

Several options are provided to model heat transfer between the condensed-phase and the gas-phase inside a decomposing solid. It is often assumed that the gas phase and the condensed-phase are in thermal equilibrium due to the much larger volumetric heat capacity of the condensed-phase compared to the gas-phase. If thermal equilibrium is desired, then set THERMAL_EQUILIBRIUM = .TRUE.. In thermal equilibrium mode, two options are provided for calculating the heat transfer between the gaseous and condensed phases, see the discussion surrounding Equation 70 in the Gpyro Technical Reference [1] for details. The default approach assumes $\rho_g \overline{\psi} c_{pg} << \overline{\rho c}$ and $\overline{\psi} k_g << \bar{k}$; if these approximations cannot be made, then set FULL_QSG = .TRUE.. Note that if the user sets both SOLVE_GAS_ENERGY = .TRUE. and THERMAL_EQUILIBRIUM = .TRUE., any heat release from homogeneous gas phase reactions is added to the condensed phase (via $\dot{Q}'''_{s-g}$) rather than the gas phase.

If SOLVE_GAS_ENERGY = .TRUE. and THERMAL_EQUILIBRIUM = .FALSE., a "two temperature" simulation results, wherein the temperature of the gas-phase and the condensed-

phase are not necessarily the same at a given spatial location. This can increase the number of iterations required for convergence, thereby increasing the required CPU time. Note that with SOLVE_GAS_ENERGY = .TRUE. and THERMAL_EQUILIBRIUM = .FALSE., any heat release from homogeneous gas phase reactions is distributed to the gas phase. For two-temperature simulations, the heat transfer rate between the condensed-phase and the gas-phase is proportional to h_{cv}, the volumetric heat transfer coefficient (see Equation 70 in the Gpyro Technical Reference [1]) and the temperature difference between the two phases, i.e. $\dot{Q}'''_{s-g} = h_{cv}\left(T - T_g\right)$. In the simplest two-temperature simulation, h_{cv} is a constant. If a constant volumetric heat transfer coefficient is needed, then set the keyword HCV to a positive value. Alternatively, heat transfer between the condensed-phase and the gas-phase can be omitted altogether by specifying HCV = 0 (this is the assumption in the FDS5 pyrolysis model).

Alternatively, h_{cv} can be calculated from a Nusselt number correlation of the form:

$\mathrm{Nu} = a + b\ \mathrm{Re}^c = h_{cv}\ d_p^2 / k_g$

If it is desired to use a correlation of this form to calculate h_{cv}, then set HCV equal to a negative number and specify the values of a, b, and c in the above equation (respectively, NU_A, NU_B, and NU_C).

12.3 Iterations and convergence

Several parameters affect convergence/divergence. NTDMA_ITERATIONS is the maximum number of iterations that Gpyro will take in a given time step in an attempt to obtain a converged solution (based on the convergence criteria specified elsewhere). If Gpyro reaches this number of iterations and the solution hasn't yet converged, then Gpyro will decrease the time step and try again. The parameters NSSPECIESITERNS and NCONTINUITYITERNS are integers that control the number of sub-iterations for the condensed phase species and mass conservation equations, respectively. This is necessary because the way that the condensed phase mass and species conservation equations are solved numerically, the unknown value at the next time step depends on itself so local iterations can sometimes promote convergence. However, both NSSPECITERNS and NCONTINUITYITERNS are normally set to 1. The parameter ALPHA is the relaxation factor. When it is set to 1, no relaxation is used; but for values of ALPHA between 0 and 1, solution relaxation is implemented according to Equation 140 in the Gpyro Technical Reference [1]. Currently, the same relaxation parameter is applied to all equations, although there is really no reason (other than convenience) to use the same relaxation parameter for all equations.

Convergence criteria are specified via the keywords TMPTOL, HTOL, YITOL, PTOL, YJTOL, and HGTOL. The parameter TMPTOL is the absolute convergence criterion for the condensed-phase energy equation, *i.e.* the condensed-phase energy equation is converged when the temperature changes by no more than TMPTOL (units of K) in any cell between iterations. Similarly, PTOL (units of Pa) is the absolute convergence criterion for the pressure evolution equation. YITOL is the relative convergence criterion for the condensed phase species conservation equations. HTOL is slightly different; it is the convergence criterion for the Newton iteration used to extract temperature from weighted enthalpy. That is, the local temperature in a cell is "found" when the

difference between the enthalpy calculated from the energy conservation equation and the weighted enthalpy calculated from the found temperature match within HTOL. It may be helpful to review the discussion of Newton iteration in Section 6.8 of the Gpyro Technical Reference [1] if this is not clear. The parameter YJTOL is the relative convergence criterion for the gaseous species conservation equations. HGTOL is the absolute convergence criterion for the gas phase energy equation (in units of J/kg), *i.e.* the gas phase energy equation is converged when the solution in all cells changes by no more than HGTOL between iterations. The user is strongly encouraged to check that the user-specified tolerance has no effect on the converged solution, i.e. the solution does not change as all tolerances are decreased by an order of magnitude. Particularly for long-duration, low heating rate TGA/DSC simulations, and for a non-unity value of the global relaxation parameter α, numerical errors can accumulate over time and tighter tolerances than the default values may be necessary.

If EXPLICIT_T = .TRUE. then Gpyro uses the converged temperature from the previous time step to calculate reaction rates and the thermal properties for the current time step. The advantage of doing this is that the strong temperature dependency of reaction rates (and secondarily, thermal properties) does not hinder convergence because the reaction rates are not changing from iteration to iteration due to the newly-calculated temperature profile. This can also reduce the required CPU time because the temperature part of the source terms must be calculated only once per time step rather than at every iteration. However, since Gpyro is formulated as a fully implicit code, technically the temperature at the current iteration should be used to evaluate the reaction rates and the thermal properties, so EXPLICIT_T should normally be set to .FALSE.. Doing so preserves the correct long-time (or steady-state) behavior.

12.3 Equations to solve

Setting SOLVE_GAS_YJ = .TRUE. tells Gpyro to solve the gas phase species conservation equation.

If the logical parameter SOLVE_GAS_ENERGY is set to .TRUE., then Gpyro will solve the gas phase energy equation (in this case, the user should also set SOLVE_GAS_YJ to .TRUE.).

The logical parameter SOLVE_PRESSURE tells Gpyro whether or not to solve the pressure (momentum) equation. If SOLVE_PRESSURE = .FALSE., then Gpyro uses gaseous mass conservation to calculate the local convective mass flux from Equation 143 of the Gpyro Technical Reference [1]; otherwise it is determined from Darcy's law and the calculated pressure gradient.

The flag SOLVE_POROSITY tells Gpyro whether or not to solve a separate equation to determine the porosity.

12.4 Parameters affecting required CPU time

The logical parameter PROPERTY_LINTERP controls whether or not Gpyro uses a lookup table in conjunction with linear interpolation to obtain the temperature-dependent thermophysical properties or whether they are calculated explicitly, e.g. from the relation $\phi(T) = \phi_0 (T/T_r)^{n_\phi}$. If

PROPERTY_LINTERP is set to .TRUE., then a lookup table with linear interpolation is used; this requires considerably less CPU intensive than evaluating the exponential terms in every cell at every time step.

The logical parameter SHYI_CORRECTION controls whether the summation term in Equations 154a and 154b of the Gpyro Technical Reference [1]. Technically, this term is not really a "correction" but rather part of the proper definition of the conductive heat flux when it is expressed as an enthalpy gradient, so SHYI_CORRECTION should always be set to .TRUE., except for certain circumstances wherein the summation term in 134a and 134b of the Gpyro Technical Reference [1] is known to add to zero. However, this term is expensive to calculate, sometimes slows down convergence, and may not have a significant impact on the overall results. For this reason, an option was added to disable this term.

The parameter NCOEFF_UPDATE_SKIP tells Gpyro how frequently to update coefficients (thermal conductivities, specific heat capacities, etc.). If it is set to 1, then the coefficients are updated every iteration, if it is set to 5, they are updated every fifth iteration, and so on.

The logical parameter USE_TOFH_NEWTON tells Gpyro whether or not to use Newton iteration to extract temperature from the weighted enthalpy and the condensed phase mass fractions. If it is set to .FALSE., then Gpyro assumes that the specific heat capacity of each species does not depend on temperature and uses the explicit relation given in Equation 166 of the Gpyro Technical Reference [1] to calculate the temperature from the enthalpy because this is much faster than using Newton iteration. If USE_TOFH_NEWTON = .TRUE., then Newton iteration is used to solve Equation 164 of the Gpyro Technical Reference [1] for T.

12.5 Other parameters

Since Gpyro is generalized, it can be configured to solve the same equations as the FDS 5 pyrolysis model solves. To do this, set FDSMODE = .TRUE.. Gpyro contains output routines to write FDS 5 &MATL, &SURF, and &RAMP (for temperature-dependent material properties) lines that can be copied and pasted into FDS input files; these are executed only when FDSMODE = .TRUE.. It is the user's responsibility to confirm that the Gpyro calculations match the FDS calculations for the particular case under consideration. Informal comparisons show that Gpyro matches FDS5 Release Candidate 8, but not FDS5 5.1.6. The reason for this is not known and is being investigated.

If CONVENTIONAL_RXN_ORDER = .TRUE., Gpyro will use the conventional reaction order approach wherein $\left(\overline{\rho} Y_i \Delta z\right)_\Sigma$ in Equation 42 of the Gpyro Technical Reference [1] is replaced with $\left(\overline{\rho}\Delta z\right)\big|_{t=0}$.

The user should set TWOD = .TRUE. for 2D simulations. If NBC (specified in &GPYRO_QE) is greater than 1 and TWOD = .FALSE., then Gpyro will conduct multiple 1D calculations, i.e. there is no heat transfer, convection, or diffusion in the x direction.

If the user specifies USE_BC_PATCHES = .TRUE., boundary conditions are specified in the Namelist groups &GPYRO_SEBCPATCH, &GPYRO_GMBCPATCH, &GPYRO_GSBCPATCH, and &GPYRO_GEBCPATCH instead of the &GPYRO_QE Namelist group (for 1D simulations). For 2D simulations, USE_BC_PATCHES should always be set to .TRUE..

By setting NOCONSUMPTION=.TRUE., the consumption of condensed phase reactants is prevented. This has been used to eliminate surface regression in 2D flame spread simulations (in FDS) over thermoplastics when it was desirable to include 2D heat conduction. Related to surface regression, EPS is the cell size (in m) below which reactant consumption is no longer calculated. When a cell becomes very small due to reactant consumption, the absolute amount of gaseous fuel production in that cell also becomes very small. Preventing further reactions can reduce convergence problems that may be encountered as grid cells become very small. A value of 10^{-10} m (0.1 nm, smaller than the hard-sphere diameter of a single atom) was found to work well for many simulations.

If the gas phase species conservation equation is solved, the logical variable GAS_DIFFUSION tells Gpyro whether or not to include diffusive terms in the gaseous species and energy conservation equations. Under some circumstances (high Peclet numbers), it may be sufficient to consider only the convective terms in the gaseous species conservation equations, but it was found that this can leads to convergence problems. The logical parameter FRONT_GAS_DIFFUSION controls whether or not gases from the ambient diffuse into the decomposing solid when USE_BC_PATCHES = .FALSE.. If the parameter BLOWING is set to .TRUE. and USE_BC_PATCHES = .FALSE., then Gpyro uses a Couette flow approximation (Equation 110 of the Gpyro Technical Reference [1]) to account for the effect of blowing on the front face convective heat transfer coefficient; otherwise it is constant.

MINIMUM_CONDUCTIVITY specifies the lower bound of the condensed-phase thermal conductivity (in units of W/m-K) when SOLVE_POROSITY = .TRUE.. This is needed because the condensed-phase thermal conductivity approaches zero as the porosity approaches 1, and MINIMUM_CONDUCTIVITY is a temporary fix to prevent this from causing numerical problems.

See Section 8.0 for a description of CONSTANT_DHVOL.

See Section 12.2 for a description of FULL_QSG.

The parameter GASES_PRODUCED_AT_TSOLID affects (only for the two-temperature formulation) whether or not the term $\sum_{j=1}^{N}\left(\left(\dot{\omega}_{fj}''' - \dot{\omega}_{dj}'''\right)h_{g,j}\Delta z\right)_P - \left(\dot{\omega}_{fg}''' h_g \Delta z\right)_P$ is explicitly included in the gas-phase energy equation. If the approximation is made that gases generated by volatilizing the condensed-phase are produced at the gas-phase temperature, then this term is identically equal to 0. By default, this approximation (gases produced at temperature of condensed phase) is made in Gpyro. However, if the user specifies GASES_PRODUCED_AT_TSOLID = .TRUE., this approximation is not made and this term is included in the gas-phase energy equation. For more detail, see Equations 167a and 167b in the Gpyro Technical Reference [1].

13.0 MATERIAL PROPERTY ESTIMATION

Gpyro is distributed with a companion material property estimation program (see Section 2.5) that is based on genetic algorithm optimization. It is recommended that the spreadsheet-based front end be used when running the material property estimation program because it is easy to make mistakes when manually setting up text files due to the (usually) large number of parameters to be optimized. Therefore, the discussion in this section is focused around the spreadsheet-based front end rather than the actual Namelist groups and keywords that are written to an input file by the spreadsheet-based front end.

All worksheets necessary to use the genetic algorithm property estimation program are contained in the spreadsheet-based front end. There are three worksheets where the user specifies relevant parameters for the property estimation process: GA_GenInput, GA_phi, and GA_Vars. Each of these worksheets is explained below. Additionally, there are a number of worksheets in which experimental data must be placed. They are named 01, 02, 03 for experiment 1, 2, 3, and so on. The experimental data in these sheets must be laid out in a specific way, as will be described in detail below.

The user is cautioned that there is even less error checking in the Gpryo property estimation program than in the standalone Gpyro program and the inputs are somewhat more cryptic than for the standalone Gpyro program. Thus, it is easy to make a mistake during data entry, and the user should carefully check the model output to make sure the code is doing what the user intended. Often, runtime errors result from erroneous data entry, but the actual cause can be difficult to track down, even when the code is run from within a debugger.

The Gpyro property estimation program can be run either in serial or in parallel, and the serial executable file is the same as the parallel executable file. Parallelization is achieved using Message Passing Interface. The Gpyro property estimation program has been successfully run in parallel on more than 100 cores.

The following command executes the Gpyro property estimation program on a single processor:

```
gpyro_propest_x.yyy
```

Since the input file name must be `gpyro.data`, it is not necessary to specify an input file name as with the standalone Gpyro program. The name of the input file is hardcoded to avoid difficulties encountered with MPI i/o.

The parallel version has not been tested under Windows, but under Linux it is launched by issuing a command similar to the following:

```
mpirun –np 32 –machinefile machines.linux –machinedir ./
/home3/clauten/bin/gpyro_propest_x.yyy &
```

The syntax may be slightly different with LAM MPI instead of MPICH MPI, but this example tells the code to run on 32 cores (`–np 32`), look up the machine names to run on in the file

macines.linux (`-machinefile machines.linux`) located in the current directory (`-machinedir ./`), and execute the binary file called `gpyro_propest_x.yyy` located in the `/home3/clauten/bin/`.

The Gpyro property estimation program can be halted gracefully (important when running in parallel across multiple nodes) by creating a file called "stop.stop" in the working directory. The Gpyro property estimation program only checks for the existence of this file at the completion of a generation, so it may take several minutes (or even several hours) to shutdown after the file stop.stop is created. The contents of stop.stop are irrelevant; the program merely checks to see if it exists, and shuts down gracefully if it does.

13.1 Miscellaneous parameters – GA_GenInput worksheet

General parameters related to the genetic algorithm are specified in the GA_GenInput worksheet. The parameter `NGEN` is the number of generations, or evolution steps, that will be taken (unless the user halts execution prematurely). `NINDIV` is the number of individuals (or chromosomes) in a population. These concepts are explained in Section 7.2.2 of the Gpyro Technical Reference [1]. The parameter `MAXCOPIES` is the target (maximum) selection number described in Section 7.2.3 of the Gpyro Technical Reference [1].

If the code is run under Linux, then set `LINUX = .TRUE.`. This parameter isn't very important, all it does is delete the stop file which tells the code to shutdown gracefully by issuing the shell command "rm -f stop.stop" under Linux and "del stop.stop" under DOS.

The parameter `SIMULATED_EXPERIMENTAL_DATA` should normally be set to `.FALSE.`. If it is set to .TRUE., then the Gpyro property estimation program generates a synthetic set of experimental data (instead of reading in actual experimental data) using the material properties specified elsewhere. This was used during the development process to evaluate how well the genetic algorithm could perform for idealized scenarios, i.e. infinitely accurate experimental measurements with a computational model that is a perfect representation of the physical processes occurring.

`FITMIN` and `FITCLIP` are two parameters that come in to play only when an individual has a very bad fitness, or the solution does not converge. Normally, these are set to zero. The variable `FITEXPONENT` is the parameter ζ in Equation 196 of the Gpyro Technical Reference [1]. The last two parameters affect reproduction (whereby offspring are generated from parents). Normally, offspring are generated as linear combinations of parents (Equation 201 of the Gpyro Technical Reference [1]). However, under some circumstances it may be useful to use the value of only one of the parents, *i.e.* an entire gene is used. This is analogous to setting r_j^i in Equation 201 of the Gpyro Technical Reference [1] to 1.0. The fraction of genes for which this is done is `WHOLEGENEFRAC`.

The parameter `BRUTE_FORCE` indicates whether genes for the offspring should be generated randomly, rather than through the normal evolution process. Setting this parameter to `.TRUE.`

essentially disables the genetic algorithm's ability to exploit promising solutions and the code reverts to a brute force search technique.

During the search process, some solutions that do not converge for the specified initial time step may be encountered. The code can either reduce the time step until a converged solution is obtained (as would be done for a standalone simulation) or immediately kill the trial solution that did not converge. This is controlled by the parameter `KILL_NONCONVERGED_SOLUTIONS`. It is recommended that this parameter be set to `.TRUE.` because if it is set to `.FALSE.`, then some simulations that do not converge could require a very small time step and consume a large amount of CPU time, while other processors sit idle (if running in parallel)

13.2 Fitness metric weightings – GA_phi worksheet

The fitness metric weightings, i.e. the ϕ values in Equation 197 of the Gpyro Technical Reference [1], are specified in the worksheet GA_phi. Each ϕ value multiplies a fitness function (Equation 196 of the Gpyro Technical Reference [1]) that quantifies the level of agreement between model predictions and experimental data. In this way, certain experiments or measurements can be given greater importance than others. The user should first specify the number of ϕ values (the number of experimental measurements) in cell A1, labeled `NGA_PHI`.

In addition to the actual ϕ value (specified in column F) a few additional parameters must be given. The integer IQE (index of qe) in column B corresponds to the case # listed in the qe worksheet. For example, if IQE = 2, then the ϕ value specified is for case #2 in the qe worksheet.

Several types of experimental data can be used as fitness metrics: cumulative mass loss, mass loss rate, temperature, and thickness. The code is told what type of experimental data the ϕ value corresponds to in column C (TYPE). Here, the user enters one of four 3-character strings to indicate which type experimental data is being used: TMP (temperature, ºC), MLR (mass loss rate, g/m2–s), DLT (delta, or thickness, m), and CML (cumulative mass loss, g/m^2).

Column D is labeled "z_T". This column tells the code the depth (location) of a particular temperature measurement. For example, $z_T = 0$ indicates a surface temperature measurement, $z_T = 0.005$ indicates an in–depth temperature measurement 5 mm below the surface, and so on. Column D has meaning only if TMP is specified in column C.

Column E is labeled t_{stop}. This is a time in seconds that can be less than or equal to the simulation time specified in the qe worksheet. Setting a value of t_{stop} less than the simulation time directs Gpyro to calculate the fitness metric only through a time of t_{stop}. This is convenient if the surface temperature was measured but deemed accurate only prior to ignition (at which point flaming flame can add uncertainty to temperatures measured via thermocouple or optical methods). By setting the value of t_{stop} for the ϕ value corresponding to the surface temperature to equal the observed ignition time, the code will only use the measured surface temperature before the ignition time when evaluating that part of the overall fitness. The actual ϕ value is specified in column F. There are no restrictions on the range of acceptable ϕ values, except that it doesn't make sense to have a negative ϕ value. What matters is the relative values of different ϕ values.

Finally, column G is a small number (ε) that prevents the fitness metric from approaching infinity when the model calculation is very close to the experimental data point. See Equation 196 in the Gpyro Technical Reference [1] for further explanation.

13.3 Variables to optimize – GA_Vars worksheet

The model input parameters to be estimated from the available experimental data are specified in the GA_Vars worksheet. Each variable to be optimized is called a gene, and the number of variables to be optimized is specified in cell A1.

Each variable (gene) should be numbered sequentially in column A starting in row 11. In Column B, the name of the worksheet containing the variable to be optimized is specified. The valid choices are indicated in column B, rows 3–8. When optimizing variables from the gyields or hgyields worksheets, the code performs normalization to ensure that the sum of all gyields add to 1 and that the sum of all hgyields add to 0, as is required for mass conservation. However, for this to work correctly, the user must include all nonzero yields as variables to be optimized. For example, if the column of the gyields matrix for a particular condensed phase reaction has four nonzero yields, then these four yields should be specified as variables to be optimized in the GA_Vars worksheet. The homogeneous gaseous yields are similar, except that it is not necessary to specify the yield of reactant 1 because it is always -1.

Column E is the variable (gene) type. The valid variable types for each sheet name are indicated in Column E from rows 3 to 9. For example, if sprops is specified in column B, then the valid variable types are `K0Z` (thermal conductivity at reference temperature), `NKZ` (thermal conductivity exponent), and so on. If gyields or hgyields is specified in column B, then it is not necessary to specify a variable type because the code automatically knows that gene must correspond to a species yield.

Two indices are entered in columns D and E (called, very descriptively, Index 1 and Index 2). The role of these two integers depends on the sheet name specified in column B. For example, if the sheet name in column B is sprops, then column C (Index 1) is ISSPEC, the index of solid species as indicated in the sprops sheet. Similarly, if the sheet name in column B is rxns, then column C is IRXN, the index of the reaction (specified in the rxns worksheet). Index 2 does not matter for variables from worksheets sprops or gprops. However, for variables from worksheets gyields and hgyields, Index 1 is the index of the gaseous species, and Index 2 is the reaction number. Thus, if gyields is specified in column B, Index 1 is set to 2, and Index 2 is set to 3, then the gene corresponds to the gaseous yield for the formation of gaseous species 2 from heterogeneous reaction 3.

Columns F and G (MinVal and MaxVal) are the lower and upper bounds of each variable, i.e. the values of $a_{j,min}$ and $a_{j,max}$ in Equation 195 of the Gpyro Technical Reference [1]. These should be specified to limit the search space to physically realistic values, i.e. it is known that the thermal conductivity of wood isn't 1000 W/m-K. Column H is a logical parameter indicating whether the values of MinVal and MaxVal are the logarithm of the gene. For example, with a MinVal of 10 and a MaxVal of 12, if log is set to .TRUE. then the code will constrain the search from a minimum

value 10^{10} to a maximum value of 10^{12}; but if log is set to .FALSE., then the code will search for values between 10 and 12.

The mutation probability (see Section 7.2.5 of the Gpyro Technical Reference [1]) is specified in column I. The parameter v_{mut} in Equation 202b of the Gpyro Technical Reference [1] is specified in column J.

The integer in column K provides a convenient means to treat two genes as one parameter for reproduction purposes. For example, the logarithm of the pre-exponential factor and the activation energy are positively correlated, so it may be convenient to form offspring from two parents by applying the same weightings to both the log of the pre-exponential factor and the activation energy. This can be enabled by setting the value in column K to an integer greater than zero. Then, for reproduction purposes, the linear combination applied to the index of the gene specified in column K will also be applied to the current gene.

For solid properties, Index 2 is normally set to 0. However, if Index 2 is set to a solid species index (as specified in sprops) and both MinVal and MaxVal are negative, then the variable is actually the negative of the gene multiplied by the value of the analogous variable of the solid species corresponding to Index 2. This is confusing and is best illustrated by an example: if Index 2 is set to 1, MinVal is set to -1, and MaxVal is set to -0.5, then the gene search space is between 0.5 and 1.0 multiplied by the value of the analogous gene for species 1. The same also applies to the rxns worksheet. Representing the genes in this way is convenient when it is known that one variable must be smaller or larger than another; for example, the thermal conductivity of a char is likely lower than the thermal conductivity of the virgin material from which it was formed, and the activation energy of an oxidative reaction may be lower than the activation energy of an analogous anaerobic reaction.

13.4 Experimental data – 01, 02, 03, etc. worksheets

The experimental data must be pasted into worksheet 01 for case # (or IQE) 1, worksheet 02 for case # (or IQE) 2, and so on. Each ϕ value specified in GA_phi should have an analogous set of experimental data (with the exception of cumulative mass loss, which Gpyro automatically calculates by integrating the mass loss rate). The units for temperature measurements are ºC, mass loss rate is g/m^2-s, and thickness is m. For a TGA experiment, mass loss rate data should be entered as $1000 \times \mathrm{d}(m/m_0)/\mathrm{d}t$ where m is the sample mass and m_0 is the initial sample mass. Be sure to express this derivative as d/dt and not d/dT (time derivative instead of temperature derivative).

The experimental data must be entered according to a specific layout, and in an order that corresponds to the order in which the ϕ coefficients are specified in the GA_phi worksheet. Column A is the time axis for the first experimental measurement for which a ϕ value was specified for that IQE in GA_phi. Column B is the corresponding experimental data at each time specified in column A. Column C is blank, and the next set of experimental measurements start in column D, i.e. column D is the time axis for the second experimental measurement, and column E is the analogous experimental data at each time. Column F is blank, and columns G and H would contain the next set of experimental data. Column I is blank, and columns J and K would contain the next set of experimental data, and so on. The order in which the experimental data is specified must

correspond to the TYPE order specified in GA_phi. Do not specify experimental data for any cumulative mass loss measurements because this is calculated by integration from the mass loss rate. It is easy to make mistakes in this part of the data input process, and the input routines haven't been rigorously exercised, so be sure to check the output data to make sure that what the code has read in what you intended!

14.0 SAMPLES

An effective way to learn how Gpyro works is to run the samples that are included in the "samples" folder in the gpyro_x.yyy.zip archive that can be downloaded from Google Code (see Section 2.1). The samples that are included with Gpyro are summarized in Table 1. Each sample file includes the spreadsheet-based front-end file and the text output file that was generated by the front end. These input files have not been cleaned up to make them more readable.

Table 2. Gpyro sample calculations.

Name	Type	Description
01-dtg	standalone (0D)	Calculation of DTG signals for solid w/ 2-step kinetics heated at 4 different rates.
02-thermoplastic	standalone (1D)	Thermally irradiated thermoplastic including surface regression. Single step reaction.
03-char	standalone (1D)	Thermally irradiated charring solid. Multi-step reaction. 2 heat flux levels.
04-intumescent	standalone (1D)	Heating and swelling of thermally irradiated intumescent coating.
05-smolder	standalone (1D)	Opposed smolder propagation in porous medium. Demonstrates BC patches.
06-oxidative_pyrolysis	standalone (1D, multiple O_2 levels)	Thermally-irradiated charring solid w/ oxidative pyrolysis (nitrogen, air).
07-ignition_delay	standalone (1D, multiple heat fluxes)	Ignition delay curve calculation using critical surface temperature and mass loss rate as ignition criteria.
08-2d_inert	standalone (2D)	Heat conduction in 2D inert parallelepiped. Illustrates use of boundary condition patches.
09-dtg_ga	property estimation (0D)	Reaction kinetics estimation using synthetic experimental data generated from 01 above.
10-char_GA	property estimation (1D)	Material property estimation using synthetic experimental date generated from 03 above.
11-fds6_gpyro_coupled	Coupled Gpyro/FDS (3D)	Combustion of a wooden crib.

15.0 REFERENCES

[1] Lautenberger, C., "Gpyro – A Generalized Pyrolysis Model for Combustible Solids – Technical Reference Guide," Version 0.800, May 1, 2014.

[2] Lautenberger, C.W., "A Generalized Pyrolysis Model for Combustible Solids," Ph.D. Dissertation, Department of Mechanical Engineering, University of California, Berkeley, Fall 2007. http://reaxengineering.com/docs/lautenberger_dissertation.pdf

[3] Lautenberger, C. & Fernandez-Pello, A.C., "Generalized Pyrolysis Model for Combustible Solids," *Fire Safety Journal* **44** 819-839 (2009).

[4] Lautenberger, C. & Fernandez-Pello, A.C., "A Model for the Oxidative Pyrolysis of Wood," *Combustion and Flame* **156** 1503-1513 (2009).

[5] Dodd, A.B., Lautenberger, C. & Fernandez-Pello, A.C., "Numerical Examination of Two-Dimensional Smolder Structure in Polyurethane Foam," *Proceedings of the Combustion Institute* **32**: 2497-2504 (2009).

[6] Lautenberger, C., Kim, E., Dembsey, N. & Fernandez-Pello, C., "The Role of Decomposition Kinetics in Pyrolysis Modeling – Application to a Fire Retardant Polyester Composite," *Fire Safety Science* **9**: 1201-1212 (2008).

[7] Lautenberger, C., Rein, G. & Fernandez-Pello, A.C., "Application of a Genetic Algorithm to Estimate Material Properties for Fire Modeling from Bench-Scale Fire Test Data," *Fire Safety Journal* **41** 204-214 (2006)

[8] Lautenberger, C. & Fernandez-Pello, C., "Optimization Algorithms for Material Pyrolysis Property Estimation," *Fire Safety Science* **10** 751-764 (2011).

[9] Lautenberger, C., "Gpyro3D: A Three Dimensional Generalized Pyrolysis Model," accepted for publication in *Fire Safety Science* (IAFSS Symposium), 2014.

[10] McGrattan, K., Hostikka, S., McDermott, R., Floyd, J., Weinschenk, C., and Overholt, K., "Fire Dynamics Simulator Technical Reference Guide Volume 1: Mathematical Model," NIST Special Publication 1018, Sixth Edition, National Institute of Standards and Technology, Building and Fire Research Laboratory, Gaithersburg, MD, 2015. SVN 22343.

APPENDIX A. NAMELIST GROUP NAMES AND AVAILABLE KEYWORDS.

The entries for each Namelist group, the location of each keyword in the spreadsheet-based front end worksheet, default values, and units are described in the tables below. Each table starts on a new page, with clarifying notes given below each table where appropriate.

Table 3. &GPYRO_CASES keywords (external variables).
DEPRECATED – NEED TO UPDATE

Keyword	Location	Default value	Units	Description
NQE	Cell A1	1		Number of heat flux levels
QE(IQE)	Column B	50000	W/m^2	q_e (Applied heat flux level)
HC0(IQE)	Column C	10	W/m^2-K	h_{c0} (Front-face heat transfer coefficient at T_{ref})*
NHC(IQE)	Column D	0		n_{hc} (heat transfer coefficient exponent)*
DELTA0(IQE)	Column E	0.01	m	δ_0 (initial thickness in z-direction)
NCELL(IQE)	Column F	101		number of cells in z-direction
L(IQE)	Column G	1.0	m	L (width in x-direction)
NBC(IQE)	Column H	1		number of cells in x-direction
TIG(IQE)	Column I	3000	s	t_{ig} (ignition time)
QFL(IQE)	Column J	0	W/m^2	q_{fl} (flame heat flux)
TSTOP(IQE)	Column K	600.	s	Simulation stop time
MIG(IQE)	Column L	3.0	g/m^2-s	m_{ig} (critical mass flux at ignition)
TMPIG(IQE)	Column M	300.	°C	T_{ig} (critical surface temperature at ignition)
ZEROD(IQE)	Column N	.FALSE.		0-dimensional (lumped/TGA/DSC) simulation?
BETA(IQE)	Column O	10.	°C/min	β (TGA/DSC heating rate)
YJINF(IQE)	Column P-	1.0	-	$Y_{j,\infty}$ (ambient gaseous mass fractions)

* Front-face heat transfer coefficient varies with temperature as $h_c(T) = h_{c0}\left(T/T_{ref}\right)^{n_{hc}}$

Table 4. &GPYRO_SPROPS keywords (solid properties).

Keyword	Location	Default value	Units	Description
NSSPEC	Cell A1	1		Number of solid species
NAME(ISSPEC)	Column B	'null'		Species name
K0Z(ISSPEC)	Column C	0.2	W/m-K	k_{0z} (z-direction thermal conductivity at T_r)[1]
NKZ(ISSPEC)	Column D	0.0	-	n_{kz} (z-direction thermal conductivity exponent)
R0(ISSPEC)	Column E	1000.	kg/m^3	ρ_0 (density at T_r)[2]
NR(ISSPEC)	Column F	0.0	-	n_ρ (density exponent)
C0(ISSPEC)	Column G	1400.	J/kg-K	c_0 (specific heat capacity at T_r)
NC(ISSPEC)	Column H	0.0	-	n_c (specific heat capacity exponent)
EMIS(ISSPEC)	Column I	0.9	-	ε (emissivity)
KAPPA(ISSPEC)	Column J	9×10^9	m^{-1}	κ (radiation absorption coefficient)
TMELT(ISSPEC)	Column K	5000.	K	T_m (melt temperature)
DHMELT(ISSPEC)	Column L	0.0	J/kg	ΔH_m (latent heat of melting)
SIGMA2MELT(ISSPEC)	Column M	1.0	K^2	σ^2_m (controls Gaussian width of effective melt c_p)
GAMMA(ISSPEC)	Column N	0.0	m	γ (controls radiation heat transfer across pores)
PERMZ(ISSPEC)	Column O	1×10^{-10}	m^2	K_z (z-direction permeability)
RS0(ISSPEC)	Column P	1010.	kg/m^3	ρ_{s0} (solid density)
PORE_DIAMETER(ISSPEC)	Column Q	0.0005	m	d_p (pore diameter)
~~K0X(ISSPEC)~~	~~Column R~~	~~0.2~~	~~W/m-K~~	~~k_{0x} (x-direction thermal conductivity at T_r)~~[1]
~~NKX(ISSPEC)~~	~~Column S~~	~~0.0~~	~~-~~	~~n_{kx} (x-direction thermal conductivity exponent)~~
~~PERMX(ISSPEC)~~	~~Column T~~	~~1×10^{-10}~~	~~m^2~~	~~K_x (x-direction permeability)~~

1 Effective thermal conductivity when SOLVE_POROSITY = .FALSE. or thermal conductivity of nonporous solid when SOLVE_POROSITY = .TRUE.

2 Bulk density when SOLVE_POROSITY = .FALSE. or density of pure nonporous solid when SOLVE_POROSITY = .TRUE.

Table 5. &GPYRO_RXNS keywords (heterogeneous reaction properties).

Keyword	Location	Default value	Units	Description
NRXNS	Cell A1	1		Number of reactions
CFROM(IRXN)	Column B	'null'		Name of reactant species
CTO(IRXN)	Column C	'gases'		Name of product species ('gases' for noncharring)
Z(IRXN)	Column D	1×10^8	s^{-1}	Z (pre-exponential factor)
E(IRXN)	Column E	130.	kJ/mol	E (activation energy)
DHS(IRXN)	Column F	0	J/kg	ΔH_{sol} (solid part of heat of reaction)
DHV(IRXN)	Column G	1.1×10^6	J/kg	ΔH_{vol} (volatiles part of heat of reaction)
CHI(IRXN)	Column H	1.0	-	χ (parameter affecting swelling)
ORDER(IRXN)	Column I	1.0	-	n (reaction order in remaining solid)
ORDERO2(IRXN)	Column J	0.0	-	n_{O2} (reaction order in O2 concentration)
IKINETICMODEL(IRXN)	Column K	0		Index of kinetic model type (see below)
IO2TYPE(IRXN)	Column L	0		Index affecting oxygen reaction order[1]
M(IRXN)	Column M	0.0	-	m (additional rxn parameter, see below)
KCAT(IRXN)	Column N	0.0	-	K_{cat} (see kinetic model 9 below)
ICAT(IRXN)	Column O	0		icat (see kinetic model 9 below)

[1] For IO2TYPE = 0 reaction rate $\propto \left(1 + Y_{O_2}\right)^{n_{O2}}$; otherwise reaction rate $\propto Y_{O_2}^{n_{O2}}$

The heterogeneous reactions species yield matrix is specified via the &GPRYO_GYIELDS Namelist group. The only entry available on the &GPRYO_GYIELDS line is GYIELDS(IGSPEC, IRXN) where IRXN is the index of the heterogeneous reaction and IGSPEC is the index of the gaseous species formed or consumed by that reaction.

Table 6. &GPYRO_GPROPS keywords (gas properties).

Keyword	Location	Default value	Units	Description
NGSPEC	Cell A1	1		Number of gaseous species
IBG	Cell A2	1		Index of background species
IO2	Cell A3	-1		Index of oxygen
CPG	Cell A4	1000	J/kg-K	Gaseous specific heat capacity
NAME(IGSPEC)	Column B	'null'		Gaseous species name
YJ0(IGSPEC)	Column C	1.0	-	Initial gas-phase mass fraction
M(IGSPEC)	Column D	44.	g/mol	Molecular weight
SIGMA(IGSPEC)	Column E	5.061	Å	Lennard-Jones parameter
EPSOK(IGSPEC)	Column F	254.	K	Lennard-Jones parameter
~~B(IGSPEC)~~	~~Column G~~	~~0.0~~	~~J/kg~~	~~Char oxidation parameter (don't use)~~
~~TONSET(IGSPEC)~~	~~Column H~~	~~653.~~	~~K~~	~~Char oxidation parameter (don't use)~~
~~TSPAN(IGSPEC)~~	~~Column I~~	~~50.~~	~~K~~	~~Char oxidation parameter (don't use)~~

Table 7. &GPYRO_HGRXNS keywords (homogeneous gaseous reactions).

Keyword	Location	Default value	Units	Description
NHGRXNS	Cell A1	0		Number of homogeneous gaseous rxns
CREACTANT1(IHGRXN)	Column B	'null'		Name of reactant 1
CREACTANT2(IHGRXN)	Column C	'null'		Name of reactant 2
P(IHGRXN)	Column D	1.0	-	p (reactant 1 order)
Q(IHGRXN)	Column E	1.0	-	q (reactant 2 order)
B(IHGRXN)	Column F	0.0	-	b (temperature exponent)
Z(IHGRXN)	Column G	1×10^{10}	kg, m^3, s	Z (pre-exponential factor)
E(IHGRXN)	Column H	200	kJ/mol	E (activation energy)
DH(IHGRXN)	Column I	20,000	J/kg	ΔH (heat of reaction)

The homogeneous gaseous reactions species yield matrix is specified in a similar way via the &GPRYO_HGYIELDS Namelist group. The only entry available on the &GPRYO_HGYIELDS line is HGYIELDS(IGSPEC, IRXN) where IRXN is the index of the homogeneous gaseous reaction and IGSPEC is the index of the gaseous species formed or consumed by that reaction.

Table 8. &GPYRO_GENERAL keywords (general/miscellaneous parameters).

Keyword	Default value	Units	Description
DT0	0.02	s	Initial timestep
TAMB	300	K	Ambient temperature (T_∞)
TREF	300	K	Reference temperature (T_r)
P0	101300	Pa	Ambient pressure (P_0)
GX	0	m/s^2	x-component of gravity vector (g_x)
GZ	0	m/s^2	z-component of gravity vector (g_z)
THERMAL_EQUILIBRIUM	.TRUE.		Assume thermal equilibrium between gas and solid?
VHLC	0D0	W/m^3-K	Volumetric heat loss coefficient
HCV	0D0	W/m^3-K	Volumetric heat transfer coefficient
NU_A	2D0	-	a in the formula $Nu = a + b\ Re^c = h_{cv}\ d_p^2 / k_g$
NU_B	1D0	-	b in the formula $Nu = a + b\ Re^c = h_{cv}\ d_p^2 / k_g$
NU_C	0.5D0	-	c in the formula $Nu = a + b\ Re^c = h_{cv}\ d_p^2 / k_g$
NTDMA_ITERATIONS	100		# of iterations before reducing Δt
NSSPECIESITERNS	1		# of local iterations for solid species eqn
NCONTINUITYITERNS	1		# of local iterations for continuity equation
ALPHA	1		Global relaxation parameter, α
TMPTOL	1×10^{-4}	K	Solid temperature convergence tolerance (absolute)
HTOL	1×10^{-8}	J/kg	Newton iteration convergence tolerance (absolute)
YITOL	1×10^{-4}	-	Convergence tolerance for solid Y_i (relative)
PTOL	1×10^{-4}	Pa	Convergence tolerance for pressure (absolute)
YJTOL	1×10^{-4}	-	Convergence tolerance for gas Y_j (relative)
HGTOL	1×10^{-1}	J/kg	Convergence tolerance for gas enthalpy (absolute)
EXPLICIT_T	.FALSE.		Use T from previous timestep for rxn rates, props?
SOVE_GAS_YJ	.FALSE.		Solve gas species conservation equation?
SOLVE_GAS_ENERGY	.FALSE.		Solve gas-phase energy equation?
SOLVE_PRESSURE	.FALSE.		Use Darcy's law to solve for pressure distribution?
SOLVE_POROSITY	.FALSE.		Use gas-phase mass equation to solve for porosity?
PROPERTY_LINTERP	.TRUE.		Use lookup table w/ linear interpolation for props?
USE_TOFH_NEWTON	.TRUE.		Use Newton iteration to get T from weighted h?
SHYI_CORRECTION	.TRUE.		Include $d/dz[(k/c)\times h_i \times dY_i/dz]$ correction ?
NCOEFF_UPDATE_SKIP	1		Update eqn coefficients after this # of iterations
FDSMODE	.FALSE.		Emulate FDS5 pyrolysis model?
CONVENTIONAL_RXN_ORDER	.FALSE.		Use traditional treatment of reaction order?
TWOD	.FALSE.		Calculate 2D terms?
USE_BC_PATCHES	.FALSE.		Use boundary condition patches? (override qe)
NOCONSUMPTION	.FALSE.		Ignore reactant consumption?
EPS	1×10^{-10}	m	Cell size below which no further reactions occur
GAS_DIFFUSION	.TRUE.		Use diffusive terms in gas-phase equations?
FRONT_GAS_DIFFUSION	.TRUE.		Do gases from the ambient diffuse in?
BLOWING	.FALSE.		Use Couette flow approximation for blowing?
MINIMUM_CONDUCTIVITY	0	W/m-K	Minimum condensed-phase thermal conductivity
CONSTANT_DHVOL	.TRUE.		Use constant ΔH_{vol}? (Section 8.0)
FULL_QSG	.FALSE.		Use full expression for $\dot{Q}'''_{s-g}$? (Section 12.2)
GASES_PRODUCED_AT_TSOLID	.FALSE.		Only affects 2-temperature model (Section 12.5)

Table 9. `&GA_GENINPUT` keywords (property estimation general/miscellaneous).

Keyword	Default value	Units	Description
NINDIV	250		Population size (#of individuals)
NGEN	100		Number of generations (evolution steps)
MAXCOPIES	6		Maximum number of copies - target sel rate
LINUX	.TRUE.		Is this run under Linux?
SIMULATED_EXPERIMENTAL_DATA	.FALSE.		Generate synthetic experimental data?
RESTART	.FALSE.		Restart from file restart.restart?
FITMIN	0.0		Set very bad fitness to this
FITCLIP	0.0		If fitness is <= this or NaN, set fitness to this
FITEXPONENT	2.0		Raise fitness value to this exponent
WHOLEGENEFRAC	0.5		Fraction of genes that are not linear combins.
BRUTE_FORCE	.FALSE.		Use brute force (no genetic algorithm?)
KILL_NONCONVERGED_SOLNS	.TRUE.		Kill solutions that don’t converge?